

liblava

0.8.1

Generated by Doxygen 1.12.0



<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Class Index</b>	<b>5</b>
2.1 Class List	5
<b>3 File Index</b>	<b>11</b>
3.1 File List	11
<b>4 Class Documentation</b>	<b>15</b>
4.1 <a href="#">lava::app::about_info_setting Struct Reference</a>	15
4.1.1 Detailed Description	15
4.1.2 Member Function Documentation	15
4.1.2.1 <a href="#">all()</a>	15
4.2 <a href="#">lava::allocator Struct Reference</a>	16
4.2.1 Detailed Description	16
4.2.2 Constructor & Destructor Documentation	16
4.2.2.1 <a href="#">allocator()</a>	16
4.2.3 Member Function Documentation	17
4.2.3.1 <a href="#">create()</a>	17
4.2.3.2 <a href="#">get()</a>	17
4.2.3.3 <a href="#">make()</a>	17
4.2.3.4 <a href="#">valid()</a>	18
4.3 <a href="#">lava::app Struct Reference</a>	18
4.3.1 Detailed Description	22
4.3.2 Constructor & Destructor Documentation	22
4.3.2.1 <a href="#">app()</a> [1/2]	22
4.3.2.2 <a href="#">app()</a> [2/2]	22
4.3.3 Member Function Documentation	22
4.3.3.1 <a href="#">block_cmd()</a>	22
4.3.3.2 <a href="#">draw_about()</a>	22
4.3.3.3 <a href="#">fps_cap()</a>	23
4.3.3.4 <a href="#">get_fps_info()</a>	23
4.3.3.5 <a href="#">get_frame_counter()</a>	23
4.3.3.6 <a href="#">screenshot()</a>	23
4.3.3.7 <a href="#">setup()</a>	24
4.3.3.8 <a href="#">switch_config()</a>	24
4.3.3.9 <a href="#">triple_buffer()</a>	25
4.3.3.10 <a href="#">v_sync()</a>	25
4.4 <a href="#">lava::app_config Struct Reference</a>	25
4.4.1 Detailed Description	26
4.4.2 Member Function Documentation	27
4.4.2.1 <a href="#">get_json()</a>	27

4.4.2.2 <code>set_json()</code> . . . . .	27
4.5 <code>lava::attachment</code> Struct Reference . . . . .	27
4.5.1 Detailed Description . . . . .	28
4.5.2 Constructor & Destructor Documentation . . . . .	28
4.5.2.1 <code>attachment()</code> . . . . .	28
4.5.3 Member Function Documentation . . . . .	29
4.5.3.1 <code>get_description()</code> . . . . .	29
4.5.3.2 <code>make()</code> . . . . .	29
4.5.3.3 <code>set_final_layout()</code> . . . . .	29
4.5.3.4 <code>set_format()</code> . . . . .	29
4.5.3.5 <code>set_initial_layout()</code> . . . . .	30
4.5.3.6 <code>set_layouts()</code> . . . . .	30
4.5.3.7 <code>set_load_op()</code> . . . . .	30
4.5.3.8 <code>set_op()</code> . . . . .	30
4.5.3.9 <code>set_samples()</code> . . . . .	31
4.5.3.10 <code>set_stencil_load_op()</code> . . . . .	31
4.5.3.11 <code>set_stencil_op()</code> . . . . .	31
4.5.3.12 <code>set_stencil_store_op()</code> . . . . .	31
4.5.3.13 <code>set_store_op()</code> . . . . .	32
4.6 <code>lava::benchmark_data</code> Struct Reference . . . . .	32
4.6.1 Detailed Description . . . . .	33
4.7 <code>lava::descriptor::binding</code> Struct Reference . . . . .	33
4.7.1 Detailed Description . . . . .	33
4.7.2 Member Function Documentation . . . . .	34
4.7.2.1 <code>get()</code> . . . . .	34
4.7.2.2 <code>make()</code> . . . . .	34
4.7.2.3 <code>set()</code> . . . . .	34
4.7.2.4 <code>set_count()</code> . . . . .	34
4.7.2.5 <code>set_samplers()</code> . . . . .	35
4.7.2.6 <code>set_stage_flags()</code> . . . . .	35
4.7.2.7 <code>set_type()</code> . . . . .	35
4.8 <code>lava::block</code> Struct Reference . . . . .	36
4.8.1 Detailed Description . . . . .	37
4.8.2 Member Function Documentation . . . . .	37
4.8.2.1 <code>activated()</code> . . . . .	37
4.8.2.2 <code>add_cmd()</code> . . . . .	38
4.8.2.3 <code>add_command()</code> . . . . .	38
4.8.2.4 <code>collect_buffers()</code> . . . . .	38
4.8.2.5 <code>create()</code> . . . . .	38
4.8.2.6 <code>get_cmd_order()</code> . . . . .	39
4.8.2.7 <code>get_command_buffer()</code> [1/2] . . . . .	39
4.8.2.8 <code>get_command_buffer()</code> [2/2] . . . . .	39

4.8.2.9 <code>get_commands()</code>	40
4.8.2.10 <code>get_current_frame()</code>	40
4.8.2.11 <code>get_device()</code>	40
4.8.2.12 <code>get_frame_count()</code>	40
4.8.2.13 <code>make()</code>	40
4.8.2.14 <code>process()</code>	40
4.8.2.15 <code>remove_cmd()</code>	41
4.8.2.16 <code>remove_command()</code>	41
4.8.2.17 <code>set_active()</code>	41
4.9 <code>lava::buffer</code> Struct Reference	42
4.9.1 Detailed Description	43
4.9.2 Member Function Documentation	44
4.9.2.1 <code>create()</code>	44
4.9.2.2 <code>create_mapped()</code>	44
4.9.2.3 <code>flush()</code>	45
4.9.2.4 <code>get()</code>	45
4.9.2.5 <code>get_address()</code>	45
4.9.2.6 <code>get_allocation()</code>	46
4.9.2.7 <code>get_allocation_info()</code>	46
4.9.2.8 <code>get_descriptor_info()</code>	46
4.9.2.9 <code>get_device()</code>	46
4.9.2.10 <code>get_device_memory()</code>	46
4.9.2.11 <code>get_mapped_data()</code>	47
4.9.2.12 <code>get_size()</code>	47
4.9.2.13 <code>make()</code>	47
4.9.2.14 <code>valid()</code>	47
4.10 <code>lava::c_data</code> Struct Reference	47
4.10.1 Detailed Description	48
4.10.2 Constructor & Destructor Documentation	48
4.10.2.1 <code>c_data()</code> [1/2]	48
4.10.2.2 <code>c_data()</code> [2/2]	48
4.11 <code>lava::json_file::callback</code> Struct Reference	49
4.11.1 Detailed Description	49
4.12 <code>lava::swapchain::callback</code> Struct Reference	49
4.12.1 Detailed Description	50
4.13 <code>lava::camera</code> Struct Reference	50
4.13.1 Detailed Description	52
4.13.2 Member Function Documentation	53
4.13.2.1 <code>activated()</code>	53
4.13.2.2 <code>calc_view_projection()</code>	53
4.13.2.3 <code>create()</code>	53
4.13.2.4 <code>get_descriptor_info()</code>	53

4.13.2.5	<a href="#">get_projection()</a>	54
4.13.2.6	<a href="#">get_view()</a>	54
4.13.2.7	<a href="#">handle()</a> [1/3]	54
4.13.2.8	<a href="#">handle()</a> [2/3]	54
4.13.2.9	<a href="#">handle()</a> [3/3]	54
4.13.2.10	<a href="#">moving()</a>	55
4.13.2.11	<a href="#">set_active()</a>	55
4.13.2.12	<a href="#">set_movement_keys()</a>	55
4.13.2.13	<a href="#">update_view()</a> [1/2]	55
4.13.2.14	<a href="#">update_view()</a> [2/2]	56
4.13.2.15	<a href="#">valid()</a>	56
4.14	<a href="#">lava::command Struct Reference</a>	56
4.14.1	<a href="#">Detailed Description</a>	58
4.14.2	<a href="#">Member Function Documentation</a>	58
4.14.2.1	<a href="#">create()</a>	58
4.14.2.2	<a href="#">destroy()</a>	58
4.14.2.3	<a href="#">make()</a>	59
4.15	<a href="#">lava::compute_pipeline Struct Reference</a>	59
4.15.1	<a href="#">Detailed Description</a>	62
4.15.2	<a href="#">Member Function Documentation</a>	62
4.15.2.1	<a href="#">bind()</a>	62
4.15.2.2	<a href="#">copy_from()</a>	62
4.15.2.3	<a href="#">copy_to()</a>	62
4.15.2.4	<a href="#">get_shader_stage()</a>	62
4.15.2.5	<a href="#">make()</a>	63
4.15.2.6	<a href="#">set()</a>	64
4.15.2.7	<a href="#">set_shader_stage()</a>	64
4.16	<a href="#">lava::imgui::config Struct Reference</a>	64
4.16.1	<a href="#">Detailed Description</a>	65
4.17	<a href="#">lava::log::config Struct Reference</a>	65
4.17.1	<a href="#">Detailed Description</a>	65
4.18	<a href="#">lava::configurable Struct Reference</a>	66
4.18.1	<a href="#">Detailed Description</a>	66
4.18.2	<a href="#">Member Function Documentation</a>	66
4.18.2.1	<a href="#">get_json()</a>	66
4.18.2.2	<a href="#">set_json()</a>	66
4.19	<a href="#">lava::render_pipeline::create_info Struct Reference</a>	67
4.19.1	<a href="#">Detailed Description</a>	67
4.20	<a href="#">lava::device::create_param Struct Reference</a>	67
4.20.1	<a href="#">Detailed Description</a>	68
4.20.2	<a href="#">Member Function Documentation</a>	68
4.20.2.1	<a href="#">add_dedicated_queues()</a>	68

4.20.2.2 <code>add_queue()</code> . . . . .	69
4.20.2.3 <code>add_queues()</code> . . . . .	69
4.20.2.4 <code>verify_queues()</code> . . . . .	70
4.21 <code>lava::instance::create_param</code> Struct Reference . . . . .	70
4.21.1 Detailed Description . . . . .	70
4.22 <code>lava::data</code> Struct Reference . . . . .	71
4.22.1 Detailed Description . . . . .	72
4.22.2 Constructor & Destructor Documentation . . . . .	72
4.22.2.1 <code>data()</code> . . . . .	72
4.22.3 Member Function Documentation . . . . .	72
4.22.3.1 <code>allocate()</code> . . . . .	72
4.22.3.2 <code>as_c_ptr()</code> . . . . .	72
4.22.3.3 <code>as_ptr()</code> . . . . .	73
4.22.3.4 <code>end()</code> . . . . .	73
4.22.3.5 <code>set()</code> . . . . .	73
4.23 <code>lava::data_provider</code> Struct Reference . . . . .	74
4.23.1 Detailed Description . . . . .	74
4.24 <code>lava::instance::debug_config</code> Struct Reference . . . . .	74
4.24.1 Detailed Description . . . . .	75
4.25 <code>lava::descriptor</code> Struct Reference . . . . .	75
4.25.1 Detailed Description . . . . .	77
4.25.2 Member Function Documentation . . . . .	77
4.25.2.1 <code>add()</code> . . . . .	77
4.25.2.2 <code>add_binding()</code> . . . . .	77
4.25.2.3 <code>allocate()</code> [1/2] . . . . .	78
4.25.2.4 <code>allocate()</code> [2/2] . . . . .	78
4.25.2.5 <code>allocate_set()</code> . . . . .	78
4.25.2.6 <code>allocate_sets()</code> . . . . .	78
4.25.2.7 <code>create()</code> . . . . .	78
4.25.2.8 <code>deallocate()</code> [1/2] . . . . .	79
4.25.2.9 <code>deallocate()</code> [2/2] . . . . .	79
4.25.2.10 <code>deallocate_set()</code> . . . . .	79
4.25.2.11 <code>deallocate_sets()</code> . . . . .	79
4.25.2.12 <code>get()</code> . . . . .	80
4.25.2.13 <code>get_binding_count()</code> . . . . .	80
4.25.2.14 <code>get_bindings()</code> . . . . .	80
4.25.2.15 <code>get_device()</code> . . . . .	80
4.25.2.16 <code>make()</code> . . . . .	81
4.26 <code>lava::device</code> Struct Reference . . . . .	81
4.26.1 Detailed Description . . . . .	85
4.26.2 Member Function Documentation . . . . .	85
4.26.2.1 <code>alloc()</code> . . . . .	85

4.26.2.2	call()	85
4.26.2.3	compute_queue()	85
4.26.2.4	compute_queues()	85
4.26.2.5	create()	85
4.26.2.6	get()	86
4.26.2.7	get_allocator()	86
4.26.2.8	get_compute_queue()	86
4.26.2.9	get_compute_queues()	87
4.26.2.10	get_features()	87
4.26.2.11	get_graphics_queue()	87
4.26.2.12	get_graphics_queues()	87
4.26.2.13	get_physical_device()	88
4.26.2.14	get_properties()	88
4.26.2.15	get_queues()	88
4.26.2.16	get_transfer_queue()	88
4.26.2.17	get_transfer_queues()	89
4.26.2.18	get_vk_physical_device()	89
4.26.2.19	graphics_queue()	89
4.26.2.20	graphics_queues()	89
4.26.2.21	make()	89
4.26.2.22	queues()	89
4.26.2.23	set_allocator()	89
4.26.2.24	surface_supported()	90
4.26.2.25	transfer_queue()	90
4.26.2.26	transfer_queues()	90
4.26.2.27	wait_for_idle()	90
4.27	lava::device_table Struct Reference	91
4.27.1	Detailed Description	92
4.27.2	Member Function Documentation	92
4.27.2.1	vkAcquireNextImageKHR()	92
4.27.2.2	vkAllocateCommandBuffers() [1/2]	93
4.27.2.3	vkAllocateCommandBuffers() [2/2]	93
4.27.2.4	vkCreateCommandPool() [1/3]	93
4.27.2.5	vkCreateCommandPool() [2/3]	93
4.27.2.6	vkCreateCommandPool() [3/3]	94
4.27.2.7	vkCreateFence() [1/2]	94
4.27.2.8	vkCreateFence() [2/2]	94
4.27.2.9	vkCreateImageView() [1/2]	94
4.27.2.10	vkCreateImageView() [2/2]	95
4.27.2.11	vkCreateSampler() [1/2]	95
4.27.2.12	vkCreateSampler() [2/2]	95
4.27.2.13	vkCreateSemaphore() [1/2]	95



4.27.2.14 vkCreateSemaphore() [2/2]	96
4.27.2.15 vkCreateShaderModule() [1/2]	96
4.27.2.16 vkCreateShaderModule() [2/2]	96
4.27.2.17 vkCreateSwapchainKHR() [1/2]	96
4.27.2.18 vkCreateSwapchainKHR() [2/2]	97
4.27.2.19 vkDestroyCommandPool()	97
4.27.2.20 vkDestroyFence()	97
4.27.2.21 vkDestroyImageView()	97
4.27.2.22 vkDestroySampler()	98
4.27.2.23 vkDestroySemaphore()	98
4.27.2.24 vkDestroySwapchainKHR()	98
4.27.2.25 vkFreeCommandBuffers()	98
4.27.2.26 vkGetSwapchainImagesKHR()	99
4.27.2.27 vkQueuePresentKHR()	99
4.27.2.28 vkQueueSubmit()	99
4.27.2.29 vkResetFences()	99
4.27.2.30 vkUpdateDescriptorSets() [1/7]	100
4.27.2.31 vkUpdateDescriptorSets() [2/7]	100
4.27.2.32 vkUpdateDescriptorSets() [3/7]	100
4.27.2.33 vkUpdateDescriptorSets() [4/7]	100
4.27.2.34 vkUpdateDescriptorSets() [5/7]	101
4.27.2.35 vkUpdateDescriptorSets() [6/7]	101
4.27.2.36 vkUpdateDescriptorSets() [7/7]	101
4.27.2.37 vkWaitForFences()	101
4.28 lava::driver Struct Reference	102
4.28.1 Detailed Description	102
4.28.2 Member Function Documentation	102
4.28.2.1 add_stage()	102
4.28.2.2 get_stages()	103
4.28.2.3 instance()	103
4.28.2.4 run()	103
4.29 lava::engine Struct Reference	104
4.29.1 Detailed Description	108
4.29.2 Member Function Documentation	108
4.29.2.1 setup()	108
4.30 lava::entity Struct Reference	108
4.30.1 Detailed Description	110
4.30.2 Member Function Documentation	110
4.30.2.1 get_id()	110
4.31 lava::file Struct Reference	110
4.31.1 Detailed Description	111
4.31.2 Constructor & Destructor Documentation	111

4.31.2.1 file()	111
4.31.3 Member Function Documentation	112
4.31.3.1 get_path()	112
4.31.3.2 get_size()	112
4.31.3.3 get_type()	112
4.31.3.4 open()	112
4.31.3.5 opened()	113
4.31.3.6 read() [1/2]	113
4.31.3.7 read() [2/2]	113
4.31.3.8 seek()	113
4.31.3.9 tell()	114
4.31.3.10 writable()	114
4.31.3.11 write()	114
4.32 lava::file_data Struct Reference	115
4.32.1 Detailed Description	117
4.32.2 Constructor & Destructor Documentation	117
4.32.2.1 file_data()	117
4.33 lava::file_delete Struct Reference	117
4.33.1 Detailed Description	118
4.33.2 Constructor & Destructor Documentation	118
4.33.2.1 file_delete()	118
4.34 lava::file_system Struct Reference	118
4.34.1 Detailed Description	120
4.34.2 Member Function Documentation	120
4.34.2.1 create_folder()	120
4.34.2.2 enumerate_files()	120
4.34.2.3 exists()	120
4.34.2.4 get_app()	121
4.34.2.5 get_base_dir()	121
4.34.2.6 get_ext()	121
4.34.2.7 get_full_base_dir()	121
4.34.2.8 get_org()	122
4.34.2.9 get_pref_dir()	122
4.34.2.10 get_real_dir()	122
4.34.2.11 get_res_dir()	122
4.34.2.12 get_version()	123
4.34.2.13 initialize()	123
4.34.2.14 mount()	123
4.34.2.15 mount_base()	123
4.34.2.16 mount_res()	124
4.34.2.17 ready()	124
4.35 lava::imgui::font Struct Reference	124

4.35.1 Detailed Description	125
4.36 <code>lava::forward_shading</code> Struct Reference	125
4.36.1 Detailed Description	126
4.36.2 Member Function Documentation	126
4.36.2.1 <code>create()</code>	126
4.36.2.2 <code>get_depth_stencil()</code>	126
4.36.2.3 <code>get_pass()</code>	126
4.36.2.4 <code>get_vk_pass()</code>	127
4.37 <code>lava::frame</code> Struct Reference	127
4.37.1 Detailed Description	129
4.37.2 Constructor & Destructor Documentation	129
4.37.2.1 <code>frame()</code> [1/2]	129
4.37.2.2 <code>frame()</code> [2/2]	129
4.37.3 Member Function Documentation	129
4.37.3.1 <code>add_run()</code>	129
4.37.3.2 <code>add_run_end()</code>	130
4.37.3.3 <code>add_run_once()</code>	130
4.37.3.4 <code>get_cmd_line()</code>	130
4.37.3.5 <code>get_env()</code>	131
4.37.3.6 <code>get_name()</code>	131
4.37.3.7 <code>get_running_time()</code>	131
4.37.3.8 <code>get_running_time_sec()</code>	131
4.37.3.9 <code>ready()</code>	131
4.37.3.10 <code>remove()</code>	131
4.37.3.11 <code>run()</code>	132
4.37.3.12 <code>set_wait_for_events()</code>	132
4.37.3.13 <code>shut_down()</code>	132
4.37.3.14 <code>waiting_for_events()</code>	132
4.38 <code>lava::frame_env</code> Struct Reference	133
4.38.1 Detailed Description	133
4.38.2 Constructor & Destructor Documentation	133
4.38.2.1 <code>frame_env()</code>	133
4.39 <code>lava::gamepad</code> Struct Reference	134
4.39.1 Detailed Description	134
4.39.2 Constructor & Destructor Documentation	134
4.39.2.1 <code>gamepad()</code>	134
4.39.3 Member Function Documentation	135
4.39.3.1 <code>get_id()</code>	135
4.39.3.2 <code>get_name()</code>	135
4.39.3.3 <code>get_pad_id()</code>	135
4.39.3.4 <code>pressed()</code>	135
4.39.3.5 <code>ready()</code>	136

4.39.3.6 update()	136
4.39.3.7 value()	136
4.40 lava::gamepad_manager Struct Reference	137
4.40.1 Detailed Description	137
4.40.2 Member Function Documentation	137
4.40.2.1 add()	137
4.40.2.2 remove()	137
4.40.2.3 singleton()	138
4.41 lava::global_logger Struct Reference	138
4.41.1 Detailed Description	138
4.41.2 Member Function Documentation	139
4.41.2.1 get()	139
4.41.2.2 set()	139
4.41.2.3 singleton()	139
4.42 lava::hex_cell Struct Reference	139
4.42.1 Detailed Description	140
4.42.2 Member Function Documentation	140
4.42.2.1 add()	140
4.42.2.2 scale()	141
4.42.2.3 subtract()	141
4.42.2.4 to_pair()	141
4.43 lava::hex_fractional_cell Struct Reference	141
4.43.1 Detailed Description	142
4.44 lava::hex_grid Struct Reference	142
4.44.1 Detailed Description	142
4.44.2 Constructor & Destructor Documentation	142
4.44.2.1 hex_grid()	142
4.44.3 Member Function Documentation	143
4.44.3.1 find()	143
4.44.3.2 to_pixel()	143
4.44.3.3 update()	143
4.45 lava::hex_layout Struct Reference	144
4.45.1 Detailed Description	144
4.46 lava::hex_offset_coord Struct Reference	144
4.46.1 Detailed Description	145
4.47 lava::hex_orientation Struct Reference	145
4.47.1 Detailed Description	145
4.48 lava::hex_point Struct Reference	145
4.48.1 Detailed Description	146
4.49 lava::imgui::icon_font Struct Reference	146
4.49.1 Detailed Description	146
4.50 lava::id Struct Reference	147

4.50.1 Detailed Description	147
4.50.2 Constructor & Destructor Documentation	147
4.50.2.1 id()	147
4.50.3 Member Function Documentation	148
4.50.3.1 to_string()	148
4.50.3.2 valid()	148
4.51 <code>lava::id_listeners&lt; T &gt;</code> Struct Template Reference	148
4.51.1 Detailed Description	148
4.51.2 Member Function Documentation	149
4.51.2.1 add()	149
4.51.2.2 get_list()	149
4.51.2.3 remove()	149
4.52 <code>lava::id_registry&lt; T, Meta &gt;</code> Struct Template Reference	150
4.52.1 Detailed Description	150
4.52.2 Member Function Documentation	151
4.52.2.1 add()	151
4.52.2.2 create()	151
4.52.2.3 exists()	151
4.52.2.4 get()	151
4.52.2.5 get_all()	152
4.52.2.6 get_all_meta()	152
4.52.2.7 get_meta()	152
4.52.2.8 remove()	152
4.52.2.9 update()	153
4.53 <code>lava::ids</code> Struct Reference	153
4.53.1 Detailed Description	154
4.53.2 Member Function Documentation	154
4.53.2.1 instance()	154
4.53.2.2 next()	154
4.54 <code>lava::image</code> Struct Reference	154
4.54.1 Detailed Description	156
4.54.2 Constructor & Destructor Documentation	156
4.54.2.1 image()	156
4.54.3 Member Function Documentation	157
4.54.3.1 create()	157
4.54.3.2 destroy()	157
4.54.3.3 get()	157
4.54.3.4 get_allocation()	158
4.54.3.5 get_depth()	158
4.54.3.6 get_device()	158
4.54.3.7 get_format()	158
4.54.3.8 get_info()	158

4.54.3.9	<a href="#">get_size()</a>	159
4.54.3.10	<a href="#">get_subresource_range()</a>	159
4.54.3.11	<a href="#">get_view()</a>	159
4.54.3.12	<a href="#">get_view_info()</a>	159
4.54.3.13	<a href="#">make()</a>	159
4.54.3.14	<a href="#">set_aspect_mask()</a>	160
4.54.3.15	<a href="#">set_component()</a>	160
4.54.3.16	<a href="#">set_flags()</a>	160
4.54.3.17	<a href="#">set_layer_count()</a>	160
4.54.3.18	<a href="#">set_layout()</a>	161
4.54.3.19	<a href="#">set_level_count()</a>	161
4.54.3.20	<a href="#">set_tiling()</a>	161
4.54.3.21	<a href="#">set_usage()</a>	161
4.54.3.22	<a href="#">set_view_type()</a>	162
4.55	<a href="#">lava::image_data Struct Reference</a>	162
4.55.1	<a href="#">Detailed Description</a>	163
4.55.2	<a href="#">Member Function Documentation</a>	163
4.55.2.1	<a href="#">get_data()</a>	163
4.55.2.2	<a href="#">ready()</a>	163
4.55.2.3	<a href="#">set_data()</a>	163
4.55.2.4	<a href="#">size()</a>	164
4.56	<a href="#">lava::imgui Struct Reference</a>	164
4.56.1	<a href="#">Detailed Description</a>	166
4.56.2	<a href="#">Constructor &amp; Destructor Documentation</a>	166
4.56.2.1	<a href="#">imgui()</a>	166
4.56.3	<a href="#">Member Function Documentation</a>	166
4.56.3.1	<a href="#">activated()</a>	166
4.56.3.2	<a href="#">capture_keyboard()</a>	166
4.56.3.3	<a href="#">capture_mouse()</a>	166
4.56.3.4	<a href="#">create()</a> [1/3]	166
4.56.3.5	<a href="#">create()</a> [2/3]	167
4.56.3.6	<a href="#">create()</a> [3/3]	167
4.56.3.7	<a href="#">get_ini_file()</a>	168
4.56.3.8	<a href="#">get_input_callback()</a>	168
4.56.3.9	<a href="#">get_pipeline()</a>	168
4.56.3.10	<a href="#">ready()</a>	168
4.56.3.11	<a href="#">set_active()</a>	168
4.56.3.12	<a href="#">set_ini_file()</a>	169
4.56.3.13	<a href="#">setup()</a> [1/2]	169
4.56.3.14	<a href="#">setup()</a> [2/2]	169
4.56.3.15	<a href="#">upload_fonts()</a>	169
4.57	<a href="#">lava::input Struct Reference</a>	170

4.57.1 Detailed Description	171
4.57.2 Member Function Documentation	171
4.57.2.1 add()	171
4.57.2.2 get_mouse_position()	171
4.57.2.3 remove()	171
4.57.2.4 set_mouse_position()	171
4.58 lava::input_callback Struct Reference	172
4.58.1 Detailed Description	172
4.58.2 Member Typedef Documentation	172
4.58.2.1 func	172
4.59 lava::input_events< T > Struct Template Reference	173
4.59.1 Detailed Description	173
4.59.2 Member Function Documentation	173
4.59.2.1 add()	173
4.60 lava::instance Struct Reference	174
4.60.1 Detailed Description	175
4.60.2 Member Function Documentation	175
4.60.2.1 create()	175
4.60.2.2 get()	175
4.60.2.3 get_debug_config()	176
4.60.2.4 get_first_physical_device()	176
4.60.2.5 get_info()	176
4.60.2.6 get_physical_devices()	176
4.60.2.7 singleton()	176
4.61 lava::instance_info Struct Reference	177
4.61.1 Detailed Description	177
4.62 lava::interface Struct Reference	177
4.62.1 Detailed Description	178
4.63 lava::props::item Struct Reference	178
4.63.1 Detailed Description	179
4.63.2 Constructor & Destructor Documentation	179
4.63.2.1 item()	179
4.64 lava::json_file Struct Reference	179
4.64.1 Detailed Description	180
4.64.2 Constructor & Destructor Documentation	180
4.64.2.1 json_file()	180
4.64.3 Member Function Documentation	180
4.64.3.1 add()	180
4.64.3.2 get()	181
4.64.3.3 load()	181
4.64.3.4 remove()	181
4.64.3.5 save()	181

4.64.3.6 set()	181
4.65 lava::key_event Struct Reference	182
4.65.1 Detailed Description	183
4.65.2 Member Function Documentation	183
4.65.2.1 active()	183
4.65.2.2 pressed() [1/2]	183
4.65.2.3 pressed() [2/2]	183
4.65.2.4 released()	183
4.65.2.5 repeated()	184
4.66 lava::layer Struct Reference	184
4.66.1 Detailed Description	186
4.66.2 Constructor & Destructor Documentation	186
4.66.2.1 layer()	186
4.66.3 Member Function Documentation	186
4.66.3.1 make()	186
4.67 lava::texture::layer Struct Reference	186
4.67.1 Detailed Description	187
4.68 lava::layer_list Struct Reference	187
4.68.1 Detailed Description	188
4.68.2 Member Function Documentation	188
4.68.2.1 add() [1/2]	188
4.68.2.2 add() [2/2]	188
4.68.2.3 add_inactive()	188
4.68.2.4 get()	189
4.68.2.5 get_all()	189
4.68.2.6 remove()	189
4.69 lava::memory Struct Reference	190
4.69.1 Detailed Description	190
4.69.2 Member Function Documentation	191
4.69.2.1 alloc()	191
4.69.2.2 instance()	191
4.69.2.3 set_callbacks()	191
4.69.2.4 set_use_custom_cpu_callbacks()	191
4.70 lava::mesh_meta Struct Reference	192
4.70.1 Detailed Description	192
4.71 lava::mesh_template< T > Struct Template Reference	192
4.71.1 Detailed Description	194
4.71.2 Member Function Documentation	194
4.71.2.1 add_data()	194
4.71.2.2 bind()	194
4.71.2.3 bind_draw()	195
4.71.2.4 create()	195



4.71.2.5 draw()	195
4.71.2.6 empty()	196
4.71.2.7 get_data()	196
4.71.2.8 get_index_buffer()	196
4.71.2.9 get_indices() [1/2]	196
4.71.2.10 get_indices() [2/2]	197
4.71.2.11 get_indices_count()	197
4.71.2.12 get_vertex_buffer()	197
4.71.2.13 get_vertices() [1/2]	197
4.71.2.14 get_vertices() [2/2]	198
4.71.2.15 get_vertices_count()	198
4.71.2.16 make()	198
4.71.2.17 reload()	198
4.71.2.18 set_data()	198
4.72 lava::mesh_template_data< T > Struct Template Reference	199
4.72.1 Detailed Description	199
4.72.2 Member Function Documentation	199
4.72.2.1 move()	199
4.72.2.2 scale()	200
4.72.2.3 scale_vector()	200
4.73 lava::message_dispatcher Struct Reference	201
4.73.1 Detailed Description	201
4.73.2 Member Function Documentation	202
4.73.2.1 add_dispatch()	202
4.73.2.2 has_dispatch()	202
4.73.2.3 remove_dispatch()	202
4.73.2.4 send_message()	203
4.73.2.5 setup()	203
4.73.2.6 update()	203
4.74 lava::texture::mip_level Struct Reference	203
4.74.1 Detailed Description	204
4.75 lava::mouse_active_event Struct Reference	204
4.75.1 Detailed Description	205
4.76 lava::mouse_button_event Struct Reference	205
4.76.1 Detailed Description	205
4.76.2 Member Function Documentation	205
4.76.2.1 pressed()	205
4.76.2.2 released()	206
4.77 lava::mouse_move_event Struct Reference	206
4.77.1 Detailed Description	207
4.78 lava::mouse_position Struct Reference	207
4.78.1 Detailed Description	207

4.79	lava::no_copy_no_move Struct Reference	208
4.79.1	Detailed Description	208
4.80	lava::pair_hash Struct Reference	209
4.80.1	Detailed Description	209
4.80.2	Member Function Documentation	209
4.80.2.1	operator()	209
4.81	lava::path_drop_event Struct Reference	209
4.81.1	Detailed Description	210
4.82	lava::physical_device Struct Reference	210
4.82.1	Detailed Description	212
4.82.2	Constructor & Destructor Documentation	212
4.82.2.1	physical_device()	212
4.82.3	Member Function Documentation	212
4.82.3.1	create_default_device_param()	212
4.82.3.2	get()	212
4.82.3.3	get_device_name()	213
4.82.3.4	get_device_type_string()	213
4.82.3.5	get_driver_version()	213
4.82.3.6	get_extension_properties()	213
4.82.3.7	get_features()	213
4.82.3.8	get_memory_properties()	214
4.82.3.9	get_properties()	214
4.82.3.10	get_queue_family()	214
4.82.3.11	get_queue_family_properties()	214
4.82.3.12	initialize()	214
4.82.3.13	make()	215
4.82.3.14	supported()	215
4.82.3.15	surface_supported()	215
4.82.3.16	swapchain_supported()	216
4.83	lava::pipeline Struct Reference	216
4.83.1	Detailed Description	218
4.83.2	Constructor & Destructor Documentation	218
4.83.2.1	pipeline()	218
4.83.3	Member Function Documentation	219
4.83.3.1	activated()	219
4.83.3.2	auto_bind()	219
4.83.3.3	bind()	219
4.83.3.4	create()	219
4.83.3.5	get()	220
4.83.3.6	get_device()	220
4.83.3.7	get_layout()	220
4.83.3.8	ready()	220

4.83.3.9 set_active()	220
4.83.3.10 set_auto_bind()	221
4.83.3.11 set_layout()	221
4.83.3.12 setup()	221
4.84 lava::pipeline_layout Struct Reference	222
4.84.1 Detailed Description	223
4.84.2 Member Function Documentation	223
4.84.2.1 add() [1/2]	223
4.84.2.2 add() [2/2]	223
4.84.2.3 add_descriptor()	223
4.84.2.4 add_push_constant_range()	224
4.84.2.5 bind()	224
4.84.2.6 bind_descriptor_set()	224
4.84.2.7 create()	224
4.84.2.8 get()	225
4.84.2.9 get_descriptors()	225
4.84.2.10 get_device()	225
4.84.2.11 get_push_constant_ranges()	225
4.84.2.12 make()	226
4.85 lava::platform Struct Reference	226
4.85.1 Detailed Description	227
4.85.2 Member Function Documentation	227
4.85.2.1 create() [1/2]	227
4.85.2.2 create() [2/2]	227
4.85.2.3 create_device()	227
4.85.2.4 get_devices()	228
4.85.2.5 remove()	228
4.86 lava::descriptor::pool Struct Reference	228
4.86.1 Detailed Description	230
4.86.2 Member Function Documentation	230
4.86.2.1 create()	230
4.86.2.2 get()	230
4.86.2.3 get_device()	230
4.86.2.4 get_max()	231
4.86.2.5 get_sizes()	231
4.86.2.6 make()	231
4.87 lava::producer Struct Reference	231
4.87.1 Detailed Description	232
4.87.2 Member Function Documentation	232
4.87.2.1 add_mesh()	232
4.87.2.2 add_texture()	233
4.87.2.3 compile_shader()	233

4.87.2.4 create_mesh()	233
4.87.2.5 create_texture()	234
4.87.2.6 get_mesh()	234
4.87.2.7 get_shader()	234
4.87.2.8 get_texture()	235
4.87.2.9 reload_shader()	235
4.88 lava::props Struct Reference	236
4.88.1 Detailed Description	237
4.88.2 Member Function Documentation	237
4.88.2.1 add()	237
4.88.2.2 check()	237
4.88.2.3 empty()	237
4.88.2.4 exists()	238
4.88.2.5 get_all()	238
4.88.2.6 get_filename()	238
4.88.2.7 get_json()	239
4.88.2.8 install()	239
4.88.2.9 load()	239
4.88.2.10 load_all()	239
4.88.2.11 operator>()	239
4.88.2.12 parse()	240
4.88.2.13 remove()	240
4.88.2.14 set_filename()	240
4.88.2.15 set_json()	241
4.88.2.16 unload()	241
4.89 lava::pseudorandom_generator Struct Reference	241
4.89.1 Detailed Description	241
4.89.2 Constructor & Destructor Documentation	241
4.89.2.1 pseudorandom_generator()	241
4.89.3 Member Function Documentation	242
4.89.3.1 get()	242
4.89.3.2 set_seed()	242
4.90 lava::queue Struct Reference	242
4.90.1 Detailed Description	243
4.90.2 Member Function Documentation	243
4.90.2.1 operator<()	243
4.90.2.2 valid()	243
4.91 lava::queue_family_info Struct Reference	244
4.91.1 Detailed Description	244
4.91.2 Member Function Documentation	244
4.91.2.1 add()	244
4.91.2.2 count()	245

4.92	lava::queue_info Struct Reference	245
4.92.1	Detailed Description	245
4.93	lava::random_generator Struct Reference	245
4.93.1	Detailed Description	246
4.93.2	Member Function Documentation	246
4.93.2.1	get() [1/2]	246
4.93.2.2	get() [2/2]	246
4.94	lava::rect Struct Reference	247
4.94.1	Detailed Description	248
4.94.2	Constructor & Destructor Documentation	248
4.94.2.1	rect() [1/3]	248
4.94.2.2	rect() [2/3]	248
4.94.2.3	rect() [3/3]	248
4.94.3	Member Function Documentation	249
4.94.3.1	contains()	249
4.94.3.2	get_end_point()	249
4.94.3.3	get_origin()	249
4.94.3.4	get_size()	249
4.94.3.5	move()	249
4.94.3.6	set_size()	250
4.95	lava::render_pass Struct Reference	250
4.95.1	Detailed Description	252
4.95.2	Constructor & Destructor Documentation	252
4.95.2.1	render_pass()	252
4.95.3	Member Function Documentation	252
4.95.3.1	add() [1/4]	252
4.95.3.2	add() [2/4]	253
4.95.3.3	add() [3/4]	253
4.95.3.4	add() [4/4]	253
4.95.3.5	add_front()	253
4.95.3.6	create()	254
4.95.3.7	exists_subpass()	254
4.95.3.8	get()	254
4.95.3.9	get_clear_color()	255
4.95.3.10	get_clear_values()	255
4.95.3.11	get_device()	255
4.95.3.12	get_subpass()	255
4.95.3.13	get_subpass_count()	256
4.95.3.14	get_subpasses()	256
4.95.3.15	get_target_callback()	256
4.95.3.16	make()	256
4.95.3.17	process()	256

4.95.3.18 remove()	257
4.95.3.19 set_clear_color()	257
4.95.3.20 set_clear_values()	257
4.96 lava::render_pipeline Struct Reference	258
4.96.1 Detailed Description	262
4.96.2 Constructor & Destructor Documentation	262
4.96.2.1 render_pipeline()	262
4.96.3 Member Function Documentation	262
4.96.3.1 add()	262
4.96.3.2 add_color_blend_attachment()	263
4.96.3.3 add_dynamic_state()	263
4.96.3.4 add_shader()	263
4.96.3.5 add_shader_stage()	263
4.96.3.6 auto_line_width()	264
4.96.3.7 auto_sizing()	264
4.96.3.8 bind()	264
4.96.3.9 copy_from()	264
4.96.3.10 copy_to()	265
4.96.3.11 create()	265
4.96.3.12 get_line_width()	265
4.96.3.13 get_render_pass()	265
4.96.3.14 get_scissor()	266
4.96.3.15 get_shader_stages()	266
4.96.3.16 get_sizing()	266
4.96.3.17 get_subpass()	266
4.96.3.18 get_viewport()	266
4.96.3.19 make()	266
4.96.3.20 set()	267
4.96.3.21 set_auto_line_width()	267
4.96.3.22 set_auto_size()	267
4.96.3.23 set_depth_compare_op()	267
4.96.3.24 set_depth_test_and_write()	268
4.96.3.25 set_dynamic_states()	268
4.96.3.26 set_input_topology()	268
4.96.3.27 set_line_width() [1/2]	268
4.96.3.28 set_line_width() [2/2]	269
4.96.3.29 set_rasterization_cull_mode()	269
4.96.3.30 set_rasterization_front_face()	269
4.96.3.31 set_rasterization_polygon_mode()	269
4.96.3.32 set_render_pass()	270
4.96.3.33 set_scissor()	270
4.96.3.34 set_sizing()	270

4.96.3.35 set_subpass()	270
4.96.3.36 set_vertex_input_attribute()	271
4.96.3.37 set_vertex_input_attributes()	271
4.96.3.38 set_vertex_input_binding()	271
4.96.3.39 set_vertex_input_bindings()	271
4.96.3.40 set_viewport()	272
4.96.3.41 set_viewport_and_scissor()	272
4.97 lava::render_target Struct Reference	272
4.97.1 Detailed Description	274
4.97.2 Member Function Documentation	274
4.97.2.1 add_callback()	274
4.97.2.2 create()	275
4.97.2.3 get_backbuffer()	275
4.97.2.4 get_backbuffer_image()	275
4.97.2.5 get_backbuffers()	276
4.97.2.6 get_device()	276
4.97.2.7 get_format()	276
4.97.2.8 get_frame_count()	276
4.97.2.9 get_image()	277
4.97.2.10 get_size()	277
4.97.2.11 get_swapchain()	277
4.97.2.12 make()	277
4.97.2.13 reload_request()	277
4.97.2.14 remove_callback()	277
4.97.2.15 resize()	278
4.98 lava::renderer Struct Reference	278
4.98.1 Detailed Description	280
4.98.2 Member Function Documentation	280
4.98.2.1 begin_frame()	280
4.98.2.2 create()	280
4.98.2.3 end_frame()	280
4.98.2.4 frame()	280
4.98.2.5 get_device()	281
4.98.2.6 get_frame()	281
4.99 lava::driver::result Struct Reference	281
4.99.1 Detailed Description	282
4.100 lava::reversion_wrapper< T > Struct Template Reference	282
4.100.1 Detailed Description	282
4.101 lava::run_time Struct Reference	282
4.101.1 Detailed Description	283
4.102 lava::scoped_label< T > Struct Template Reference	283
4.102.1 Detailed Description	283

4.102.2 Constructor & Destructor Documentation	284
4.102.2.1 <code>scoped_label()</code>	284
4.103 <code>lava::scroll_event</code> Struct Reference	284
4.103.1 Detailed Description	285
4.104 <code>lava::scroll_offset</code> Struct Reference	285
4.104.1 Detailed Description	285
4.105 <code>lava::semantic_version</code> Struct Reference	285
4.105.1 Detailed Description	286
4.106 <code>lava::pipeline::shader_stage</code> Struct Reference	286
4.106.1 Detailed Description	287
4.106.2 Member Function Documentation	287
4.106.2.1 <code>add_specialization_entry()</code>	287
4.106.2.2 <code>create()</code>	287
4.106.2.3 <code>get_create_info()</code>	287
4.106.2.4 <code>make()</code>	288
4.106.2.5 <code>set_stage()</code>	289
4.107 <code>lava::stage</code> Struct Reference	289
4.107.1 Detailed Description	290
4.107.2 Constructor & Destructor Documentation	290
4.107.2.1 <code>stage()</code>	290
4.108 <code>lava::staging</code> Struct Reference	290
4.108.1 Detailed Description	291
4.108.2 Member Function Documentation	291
4.108.2.1 <code>add()</code>	291
4.108.2.2 <code>busy()</code>	291
4.108.2.3 <code>stage()</code>	291
4.109 <code>lava::window::state</code> Struct Reference	292
4.109.1 Detailed Description	292
4.110 <code>lava::subpass</code> Struct Reference	293
4.110.1 Detailed Description	295
4.110.2 Member Function Documentation	295
4.110.2.1 <code>activated()</code>	295
4.110.2.2 <code>add()</code>	295
4.110.2.3 <code>add_front()</code>	295
4.110.2.4 <code>add_preserve_attachment()</code>	295
4.110.2.5 <code>get_description()</code>	296
4.110.2.6 <code>make()</code>	296
4.110.2.7 <code>process()</code>	296
4.110.2.8 <code>remove()</code>	296
4.110.2.9 <code>set()</code>	297
4.110.2.10 <code>set_active()</code>	297
4.110.2.11 <code>set_color_attachment()</code> [1/2]	297



4.110.2.12 set_color_attachment() [2/2]	297
4.110.2.13 set_color_attachments()	298
4.110.2.14 set_depth_stencil_attachment() [1/2]	298
4.110.2.15 set_depth_stencil_attachment() [2/2]	298
4.110.2.16 set_input_attachment() [1/2]	298
4.110.2.17 set_input_attachment() [2/2]	299
4.110.2.18 set_input_attachments()	299
4.110.2.19 set_preserve_attachments()	299
4.110.2.20 set_resolve_attachment() [1/2]	299
4.110.2.21 set_resolve_attachment() [2/2]	300
4.110.2.22 set_resolve_attachments()	300
4.111 lava::subpass_dependency Struct Reference	300
4.111.1 Detailed Description	301
4.111.2 Member Function Documentation	301
4.111.2.1 get_dependency()	301
4.111.2.2 make()	302
4.111.2.3 set_access_mask()	303
4.111.2.4 set_dependency_flags()	303
4.111.2.5 set_dst_access_mask()	303
4.111.2.6 set_dst_stage_mask()	303
4.111.2.7 set_dst_subpass()	304
4.111.2.8 set_src_access_mask()	304
4.111.2.9 set_src_stage_mask()	304
4.111.2.10 set_src_subpass()	304
4.111.2.11 set_stage_mask()	305
4.111.2.12 set_subpass()	305
4.112 lava::surface_format_request Struct Reference	305
4.112.1 Detailed Description	306
4.112.2 Member Data Documentation	306
4.112.2.1 formats	306
4.113 lava::swapchain Struct Reference	306
4.113.1 Detailed Description	308
4.113.2 Member Function Documentation	308
4.113.2.1 add_callback()	308
4.113.2.2 create()	308
4.113.2.3 get()	309
4.113.2.4 get_backbuffer_count()	309
4.113.2.5 get_backbuffers()	309
4.113.2.6 get_color_space()	309
4.113.2.7 get_device()	309
4.113.2.8 get_format()	310
4.113.2.9 get_size()	310

4.113.2.10 reload_request()	310
4.113.2.11 remove_callback()	310
4.113.2.12 resize()	310
4.113.2.13 surface_supported()	311
4.113.2.14 triple_buffer()	311
4.113.2.15 v_sync()	311
4.114 lava::target_callback Struct Reference	312
4.114.1 Detailed Description	312
4.115 lava::telegram Struct Reference	312
4.115.1 Detailed Description	313
4.115.2 Constructor & Destructor Documentation	313
4.115.2.1 telegram()	313
4.115.3 Member Function Documentation	313
4.115.3.1 operator<()	313
4.115.3.2 operator==(())	314
4.116 lava::telegraph Struct Reference	314
4.116.1 Detailed Description	315
4.116.2 Member Function Documentation	315
4.116.2.1 send_message()	315
4.117 lava::texture Struct Reference	315
4.117.1 Detailed Description	317
4.117.2 Member Function Documentation	317
4.117.2.1 create()	317
4.117.2.2 get_descriptor_info()	318
4.117.2.3 get_format()	318
4.117.2.4 get_image()	318
4.117.2.5 get_size()	318
4.117.2.6 get_type()	318
4.117.2.7 make()	319
4.117.2.8 stage()	319
4.117.2.9 upload()	319
4.118 lava::texture_file Struct Reference	320
4.118.1 Detailed Description	320
4.119 lava::thread_pool Struct Reference	320
4.119.1 Detailed Description	321
4.119.2 Member Function Documentation	321
4.119.2.1 enqueue()	321
4.119.2.2 setup()	321
4.120 lava::timer Struct Reference	321
4.120.1 Detailed Description	322
4.120.2 Member Function Documentation	322
4.120.2.1 elapsed()	322

4.121	lava::tooltip Struct Reference	322
4.121.1	Detailed Description	323
4.121.2	Constructor & Destructor Documentation	323
4.121.2.1	tooltip()	323
4.122	lava::tooltip_list Struct Reference	323
4.122.1	Detailed Description	323
4.122.2	Member Function Documentation	324
4.122.2.1	add()	324
4.122.2.2	format_string()	324
4.122.2.3	get_list()	324
4.122.2.4	set()	324
4.123	lava::u_data Struct Reference	325
4.123.1	Detailed Description	326
4.123.2	Constructor & Destructor Documentation	326
4.123.2.1	u_data() [1/2]	326
4.123.2.2	u_data() [2/2]	326
4.124	lava::version Struct Reference	327
4.124.1	Detailed Description	327
4.125	lava::vertex Struct Reference	327
4.125.1	Detailed Description	328
4.125.2	Member Function Documentation	328
4.125.2.1	operator==( )	328
4.126	lava::vk_result Struct Reference	328
4.126.1	Detailed Description	329
4.126.2	Member Function Documentation	329
4.126.2.1	operator bool()	329
4.127	lava::window Struct Reference	329
4.127.1	Detailed Description	333
4.127.2	Constructor & Destructor Documentation	333
4.127.2.1	window()	333
4.127.3	Member Function Documentation	333
4.127.3.1	assign()	333
4.127.3.2	close_request()	333
4.127.3.3	create()	333
4.127.3.4	create_surface()	334
4.127.3.5	decorated()	334
4.127.3.6	detect_monitor()	334
4.127.3.7	floating()	334
4.127.3.8	focused()	335
4.127.3.9	fullscreen()	335
4.127.3.10	get()	335
4.127.3.11	get_aspect_ratio()	335

4.127.3.12	<a href="#">get_content_scale()</a>	335
4.127.3.13	<a href="#">get_framebuffer_size()</a> [1/2]	336
4.127.3.14	<a href="#">get_framebuffer_size()</a> [2/2]	336
4.127.3.15	<a href="#">get_mouse_position()</a> [1/2]	336
4.127.3.16	<a href="#">get_mouse_position()</a> [2/2]	336
4.127.3.17	<a href="#">get_position()</a>	336
4.127.3.18	<a href="#">get_save_name()</a>	337
4.127.3.19	<a href="#">get_size()</a> [1/2]	337
4.127.3.20	<a href="#">get_size()</a> [2/2]	337
4.127.3.21	<a href="#">get_state()</a>	337
4.127.3.22	<a href="#">get_title()</a>	338
4.127.3.23	<a href="#">handle_resize()</a>	338
4.127.3.24	<a href="#">hovered()</a>	338
4.127.3.25	<a href="#">iconified()</a>	338
4.127.3.26	<a href="#">maximized()</a>	338
4.127.3.27	<a href="#">resizable()</a>	339
4.127.3.28	<a href="#">resize_request()</a>	339
4.127.3.29	<a href="#">save_title()</a>	339
4.127.3.30	<a href="#">set_decorated()</a>	339
4.127.3.31	<a href="#">set_floating()</a>	339
4.127.3.32	<a href="#">set_fullscreen()</a>	340
4.127.3.33	<a href="#">set_icon()</a>	340
4.127.3.34	<a href="#">set_mouse_position()</a>	340
4.127.3.35	<a href="#">set_position()</a>	340
4.127.3.36	<a href="#">set_resizable()</a>	341
4.127.3.37	<a href="#">set_save_name()</a>	341
4.127.3.38	<a href="#">set_size()</a>	341
4.127.3.39	<a href="#">set_state()</a>	341
4.127.3.40	<a href="#">set_title()</a>	342
4.127.3.41	<a href="#">show_save_title()</a>	342
4.127.3.42	<a href="#">switch_mode()</a>	342
4.127.3.43	<a href="#">switch_mode_request()</a>	342
4.127.3.44	<a href="#">visible()</a>	343
<b>5 File Documentation</b>		<b>345</b>
5.1	<a href="#">liblava/app.hpp File Reference</a>	345
5.1.1	<a href="#">Detailed Description</a>	345
5.2	<a href="#">app.hpp</a>	345
5.3	<a href="#">liblava/app/app.hpp File Reference</a>	346
5.3.1	<a href="#">Detailed Description</a>	346
5.4	<a href="#">app.hpp</a>	346
5.5	<a href="#">liblava/app/benchmark.hpp File Reference</a>	348

5.5.1 Detailed Description . . . . .	349
5.5.2 Function Documentation . . . . .	349
5.5.2.1 benchmark() . . . . .	349
5.5.2.2 parse_benchmark() . . . . .	349
5.5.2.3 write_frames_json() . . . . .	350
5.6 benchmark.hpp . . . . .	351
5.7 liblava/app/camera.hpp File Reference . . . . .	351
5.7.1 Detailed Description . . . . .	352
5.8 camera.hpp . . . . .	352
5.9 liblava/app/config.hpp File Reference . . . . .	354
5.9.1 Detailed Description . . . . .	354
5.9.2 Function Documentation . . . . .	354
5.9.2.1 set_window_icon() . . . . .	354
5.10 config.hpp . . . . .	355
5.11 liblava/app/forward_shading.hpp File Reference . . . . .	355
5.11.1 Detailed Description . . . . .	356
5.12 forward_shading.hpp . . . . .	356
5.13 liblava/app/icon.hpp File Reference . . . . .	356
5.13.1 Detailed Description . . . . .	357
5.14 icon.hpp . . . . .	357
5.15 liblava/app/imgui.hpp File Reference . . . . .	370
5.15.1 Detailed Description . . . . .	371
5.15.2 Function Documentation . . . . .	371
5.15.2.1 imgui_left_spacing() . . . . .	371
5.15.2.2 setup_imgui_font() . . . . .	372
5.15.2.3 setup_imgui_font_icons() . . . . .	372
5.16 imgui.hpp . . . . .	372
5.17 liblava/asset.hpp File Reference . . . . .	375
5.17.1 Detailed Description . . . . .	375
5.18 asset.hpp . . . . .	375
5.19 liblava/asset/load_image.hpp File Reference . . . . .	376
5.19.1 Detailed Description . . . . .	376
5.19.2 Function Documentation . . . . .	376
5.19.2.1 load_image() [1/2] . . . . .	376
5.19.2.2 load_image() [2/2] . . . . .	376
5.20 load_image.hpp . . . . .	377
5.21 liblava/asset/load_mesh.hpp File Reference . . . . .	377
5.21.1 Detailed Description . . . . .	377
5.21.2 Function Documentation . . . . .	377
5.21.2.1 load_mesh() . . . . .	377
5.22 load_mesh.hpp . . . . .	378
5.23 liblava/asset/load_texture.hpp File Reference . . . . .	378

5.23.1 Detailed Description	378
5.23.2 Function Documentation	379
5.23.2.1 create_default_texture()	379
5.23.2.2 load_texture() [1/2]	379
5.23.2.3 load_texture() [2/2]	379
5.24 load_texture.hpp	380
5.25 liblava/asset/write_image.hpp File Reference	380
5.25.1 Detailed Description	380
5.25.2 Function Documentation	381
5.25.2.1 write_image_png()	381
5.26 write_image.hpp	381
5.27 liblava/base.hpp File Reference	381
5.27.1 Detailed Description	382
5.28 base.hpp	382
5.29 liblava/base/base.hpp File Reference	382
5.29.1 Detailed Description	385
5.29.2 Function Documentation	385
5.29.2.1 check()	385
5.29.2.2 failed()	385
5.29.2.3 to_api_version()	386
5.29.2.4 to_string()	386
5.29.2.5 to_version()	386
5.29.2.6 to_vk_version()	387
5.29.2.7 vk_version_to_string()	387
5.30 base.hpp	388
5.31 liblava/base/debug_utils.hpp File Reference	389
5.31.1 Detailed Description	392
5.31.2 Function Documentation	392
5.31.2.1 begin_label() [1/2]	392
5.31.2.2 begin_label() [2/2]	393
5.31.2.3 end_label() [1/2]	393
5.31.2.4 end_label() [2/2]	393
5.31.2.5 insert_label() [1/2]	393
5.31.2.6 insert_label() [2/2]	393
5.31.2.7 set_acceleration_structure_name()	394
5.31.2.8 set_acceleration_structure_nv_name()	394
5.31.2.9 set_acceleration_structure_nv_tag()	394
5.31.2.10 set_acceleration_structure_tag()	394
5.31.2.11 set_buffer_name()	395
5.31.2.12 set_buffer_tag()	395
5.31.2.13 set_buffer_view_name()	395
5.31.2.14 set_buffer_view_tag()	395

5.31.2.15 set_command_buffer_name()	396
5.31.2.16 set_command_buffer_tag()	396
5.31.2.17 set_command_pool_name()	396
5.31.2.18 set_command_pool_tag()	396
5.31.2.19 set_debug_report_callback_name()	397
5.31.2.20 set_debug_report_callback_tag()	397
5.31.2.21 set_debug_utils_messenger_name()	397
5.31.2.22 set_debug_utils_messenger_tag()	397
5.31.2.23 set_deferred_operation_name()	398
5.31.2.24 set_deferred_operation_tag()	398
5.31.2.25 set_descriptor_pool_name()	398
5.31.2.26 set_descriptor_pool_tag()	398
5.31.2.27 set_descriptor_set_layout_name()	399
5.31.2.28 set_descriptor_set_layout_tag()	399
5.31.2.29 set_descriptor_set_name()	399
5.31.2.30 set_descriptor_set_tag()	399
5.31.2.31 set_descriptor_update_template_name()	400
5.31.2.32 set_descriptor_update_template_tag()	400
5.31.2.33 set_device_memory_name()	400
5.31.2.34 set_device_memory_tag()	400
5.31.2.35 set_device_name()	401
5.31.2.36 set_device_tag()	401
5.31.2.37 set_display_mode_name()	401
5.31.2.38 set_display_mode_tag()	401
5.31.2.39 set_display_name()	402
5.31.2.40 set_display_tag()	402
5.31.2.41 set_event_name()	402
5.31.2.42 set_event_tag()	402
5.31.2.43 set_fence_name()	403
5.31.2.44 set_fence_tag()	403
5.31.2.45 set_framebuffer_name()	403
5.31.2.46 set_framebuffer_tag()	403
5.31.2.47 set_image_name()	404
5.31.2.48 set_image_tag()	404
5.31.2.49 set_image_view_name()	404
5.31.2.50 set_image_view_tag()	404
5.31.2.51 set_indirect_commands_layout_name()	405
5.31.2.52 set_indirect_commands_layout_tag()	405
5.31.2.53 set_instance_name()	405
5.31.2.54 set_instance_tag()	405
5.31.2.55 set_name()	406
5.31.2.56 set_object_name()	406

5.31.2.57 set_object_tag()	406
5.31.2.58 set_performance_configuration_name()	407
5.31.2.59 set_performance_configuration_tag()	407
5.31.2.60 set_physical_device_name()	407
5.31.2.61 set_physical_device_tag()	407
5.31.2.62 set_pipeline_cache_name()	408
5.31.2.63 set_pipeline_cache_tag()	408
5.31.2.64 set_pipeline_layout_name()	408
5.31.2.65 set_pipeline_layout_tag()	408
5.31.2.66 set_pipeline_name()	409
5.31.2.67 set_pipeline_tag()	409
5.31.2.68 set_private_data_slot_name()	409
5.31.2.69 set_private_data_slot_tag()	409
5.31.2.70 set_query_pool_name()	410
5.31.2.71 set_query_pool_tag()	410
5.31.2.72 set_queue_name()	410
5.31.2.73 set_queue_tag()	410
5.31.2.74 set_render_pass_name()	411
5.31.2.75 set_render_pass_tag()	411
5.31.2.76 set_sampler_name()	411
5.31.2.77 set_sampler_tag()	411
5.31.2.78 set_sampler_ycbcr_conversion_name()	412
5.31.2.79 set_sampler_ycbcr_conversion_tag()	412
5.31.2.80 set_semaphore_name()	412
5.31.2.81 set_semaphore_tag()	412
5.31.2.82 set_shader_module_name()	413
5.31.2.83 set_shader_module_tag()	413
5.31.2.84 set_surface_name()	413
5.31.2.85 set_surface_tag()	413
5.31.2.86 set_swapchain_name()	414
5.31.2.87 set_swapchain_tag()	414
5.31.2.88 set_tag()	414
5.31.2.89 set_validation_cache_name()	415
5.31.2.90 set_validation_cache_tag()	415
5.32 debug_utils.hpp	415
5.33 liblava/base/device.hpp File Reference	428
5.33.1 Detailed Description	428
5.33.2 Function Documentation	428
5.33.2.1 create_shader_module()	428
5.33.2.2 one_time_submit()	429
5.33.2.3 one_time_submit_pool()	429
5.34 device.hpp	430



5.35 liblava/base/device_table.hpp File Reference	432
5.35.1 Detailed Description	432
5.36 device_table.hpp	433
5.37 liblava/base/instance.hpp File Reference	437
5.37.1 Detailed Description	437
5.37.2 Function Documentation	437
5.37.2.1 check()	437
5.37.2.2 enumerate_extension_properties()	438
5.37.2.3 enumerate_layer_properties()	438
5.37.2.4 get_instance_version()	438
5.38 instance.hpp	439
5.39 liblava/base/memory.hpp File Reference	440
5.39.1 Detailed Description	440
5.39.2 Function Documentation	441
5.39.2.1 create_allocator()	441
5.39.2.2 find_memory_type()	441
5.39.2.3 find_memory_type_with_properties()	441
5.40 memory.hpp	442
5.41 liblava/base/physical_device.hpp File Reference	443
5.41.1 Detailed Description	443
5.42 physical_device.hpp	444
5.43 liblava/base/platform.hpp File Reference	445
5.43.1 Detailed Description	445
5.44 platform.hpp	445
5.45 liblava/base/queue.hpp File Reference	446
5.45.1 Detailed Description	447
5.45.2 Function Documentation	447
5.45.2.1 add_dedicated_queues()	447
5.45.2.2 add_queues()	447
5.45.2.3 set_all_queues()	448
5.45.2.4 set_default_queues()	448
5.45.2.5 verify_queues()	448
5.45.3 Variable Documentation	448
5.45.3.1 default_queue_flags	448
5.46 queue.hpp	449
5.47 liblava/block.hpp File Reference	450
5.47.1 Detailed Description	450
5.48 block.hpp	450
5.49 liblava/block/block.hpp File Reference	451
5.49.1 Detailed Description	451
5.50 block.hpp	451
5.51 liblava/block/attachment.hpp File Reference	453

5.51.1 Detailed Description	453
5.52 attachment.hpp	453
5.53 liblava/block/compute_pipeline.hpp File Reference	454
5.53.1 Detailed Description	455
5.54 compute_pipeline.hpp	455
5.55 liblava/block/descriptor.hpp File Reference	456
5.55.1 Detailed Description	456
5.56 descriptor.hpp	456
5.57 liblava/block/pipeline.hpp File Reference	458
5.57.1 Detailed Description	459
5.57.2 Function Documentation	459
5.57.2.1 create_pipeline_shader_stage()	459
5.58 pipeline.hpp	460
5.59 liblava/block/pipeline_layout.hpp File Reference	461
5.59.1 Detailed Description	461
5.60 pipeline_layout.hpp	462
5.61 liblava/block/render_pass.hpp File Reference	463
5.61.1 Detailed Description	463
5.62 render_pass.hpp	464
5.63 liblava/block/render_pipeline.hpp File Reference	465
5.63.1 Detailed Description	466
5.63.2 Function Documentation	466
5.63.2.1 create_pipeline_color_blend_attachment()	466
5.64 render_pipeline.hpp	466
5.65 liblava/block/subpass.hpp File Reference	469
5.65.1 Detailed Description	469
5.66 subpass.hpp	470
5.67 liblava/core.hpp File Reference	472
5.67.1 Detailed Description	472
5.68 core.hpp	472
5.69 liblava/core/data.hpp File Reference	472
5.69.1 Detailed Description	473
5.69.2 Function Documentation	473
5.69.2.1 align() [1/2]	473
5.69.2.2 align() [2/2]	474
5.69.2.3 align_up()	474
5.69.2.4 alloc_data()	474
5.69.2.5 free_data()	475
5.69.2.6 next_pow_2()	475
5.69.2.7 realloc_data()	475
5.70 data.hpp	476
5.71 liblava/core/id.hpp File Reference	478

5.71.1 Detailed Description	479
5.71.2 Function Documentation	479
5.71.2.1 add_id_map()	479
5.71.2.2 remove_id_map()	479
5.71.2.3 to_id()	480
5.72 id.hpp	480
5.73 liblava/core/misc.hpp File Reference	482
5.73.1 Detailed Description	484
5.73.2 Function Documentation	484
5.73.2.1 append()	484
5.73.2.2 begin()	484
5.73.2.3 contains()	485
5.73.2.4 end()	485
5.73.2.5 exists()	486
5.73.2.6 remove()	486
5.73.2.7 remove_chars()	486
5.73.2.8 remove_chars_copy()	487
5.73.2.9 remove_chars_if_not()	487
5.73.2.10 remove_chars_if_not_copy()	487
5.73.2.11 remove_nondigit()	488
5.73.2.12 remove_nondigit_copy()	488
5.73.2.13 remove_punctuation_marks()	488
5.73.2.14 reverse()	489
5.73.2.15 trim()	489
5.73.2.16 trim_copy()	489
5.73.2.17 trim_end()	490
5.73.2.18 trim_end_copy()	490
5.73.2.19 trim_start()	490
5.73.2.20 trim_start_copy()	490
5.74 misc.hpp	491
5.75 liblava/core/time.hpp File Reference	492
5.75.1 Detailed Description	494
5.75.2 Function Documentation	494
5.75.2.1 get_current_time()	494
5.75.2.2 get_current_timestamp()	494
5.75.2.3 get_current_timestamp_ms()	495
5.75.2.4 get_current_timestamp_us()	495
5.75.2.5 timestamp()	495
5.75.2.6 to_delta()	495
5.75.2.7 to_dt()	496
5.75.2.8 to_ms() [1/2]	496
5.75.2.9 to_ms() [2/2]	496

5.75.2.10 to_sec()	496
5.75.2.11 to_sec_fix()	497
5.76 time.hpp	497
5.77 liblava/core/types.hpp File Reference	499
5.77.1 Detailed Description	502
5.77.2 Macro Definition Documentation	502
5.77.2.1 ENUM_FLAG_OPERATORS	502
5.77.3 Typedef Documentation	503
5.77.3.1 c16	503
5.77.3.2 c32	503
5.77.3.3 c8	503
5.77.3.4 i16	503
5.77.3.5 i32	504
5.77.3.6 i64	504
5.77.3.7 i8	504
5.77.3.8 uc16	504
5.77.3.9 uc32	504
5.77.3.10 uc8	504
5.77.3.11 ui16	505
5.77.3.12 ui32	505
5.77.3.13 ui64	505
5.77.3.14 ui8	505
5.77.4 Function Documentation	505
5.77.4.1 hash_combine()	505
5.77.4.2 hash_value() [1/3]	506
5.77.4.3 hash_value() [2/3]	506
5.77.4.4 hash_value() [3/3]	506
5.77.4.5 str()	506
5.77.4.6 to_char()	507
5.77.4.7 to_i32()	507
5.77.4.8 to_i64()	507
5.77.4.9 to_index()	508
5.77.4.10 to_r32()	508
5.77.4.11 to_r64()	508
5.77.4.12 to_size_t()	509
5.77.4.13 to_ui32()	509
5.77.4.14 to_ui64()	509
5.78 types.hpp	510
5.79 liblava/core/version.hpp File Reference	513
5.79.1 Detailed Description	514
5.79.2 Function Documentation	514
5.79.2.1 to_version()	514

5.80 version.hpp . . . . .	514
5.81 liblava/engine.hpp File Reference . . . . .	515
5.81.1 Detailed Description . . . . .	515
5.82 engine.hpp . . . . .	515
5.83 liblava/engine/engine.hpp File Reference . . . . .	516
5.83.1 Detailed Description . . . . .	516
5.84 engine.hpp . . . . .	516
5.85 liblava/engine/producer.hpp File Reference . . . . .	517
5.85.1 Detailed Description . . . . .	517
5.86 producer.hpp . . . . .	518
5.87 liblava/engine/props.hpp File Reference . . . . .	519
5.87.1 Detailed Description . . . . .	519
5.88 props.hpp . . . . .	519
5.89 liblava/file.hpp File Reference . . . . .	520
5.89.1 Detailed Description . . . . .	520
5.90 file.hpp . . . . .	521
5.91 liblava/file/file.hpp File Reference . . . . .	521
5.91.1 Detailed Description . . . . .	522
5.91.2 Function Documentation . . . . .	522
5.91.2.1 file_error() . . . . .	522
5.92 file.hpp . . . . .	522
5.93 liblava/file/file_system.hpp File Reference . . . . .	523
5.93.1 Detailed Description . . . . .	524
5.94 file_system.hpp . . . . .	524
5.95 liblava/file/file_utils.hpp File Reference . . . . .	525
5.95.1 Detailed Description . . . . .	525
5.95.2 Function Documentation . . . . .	526
5.95.2.1 extension() [1/2] . . . . .	526
5.95.2.2 extension() [2/2] . . . . .	526
5.95.2.3 get_filename_from() . . . . .	526
5.95.2.4 load_file_data() . . . . .	527
5.95.2.5 read_file() . . . . .	527
5.95.2.6 remove_existing_path() . . . . .	527
5.95.2.7 write_file() . . . . .	527
5.96 file_utils.hpp . . . . .	528
5.97 json.hpp . . . . .	528
5.98 liblava/file/json_file.hpp File Reference . . . . .	529
5.98.1 Detailed Description . . . . .	529
5.99 json_file.hpp . . . . .	529
5.100 liblava/frame.hpp File Reference . . . . .	530
5.100.1 Detailed Description . . . . .	530
5.101 frame.hpp . . . . .	531

5.102 liblava/frame/frame.hpp File Reference	531
5.102.1 Detailed Description	532
5.102.2 Function Documentation	532
5.102.2.1 handle_events()	532
5.102.2.2 handle_events_timeout() [1/2]	532
5.102.2.3 handle_events_timeout() [2/2]	532
5.102.2.4 now()	533
5.103 frame.hpp	533
5.104 liblava/frame/argh.hpp File Reference	535
5.104.1 Detailed Description	536
5.104.2 Function Documentation	536
5.104.2.1 get_cmd()	536
5.104.2.2 log_command_line()	536
5.105 argh.hpp	537
5.106 liblava/frame/driver.hpp File Reference	537
5.106.1 Detailed Description	538
5.106.2 Macro Definition Documentation	538
5.106.2.1 LAVA_STAGE	538
5.106.2.2 STR	538
5.106.2.3 STR_	538
5.107 driver.hpp	539
5.108 liblava/frame/gamepad.hpp File Reference	540
5.108.1 Detailed Description	541
5.108.2 Function Documentation	541
5.108.2.1 gamepads()	541
5.109 gamepad.hpp	541
5.110 liblava/frame/input.hpp File Reference	543
5.110.1 Detailed Description	545
5.110.2 Function Documentation	546
5.110.2.1 check_mod()	546
5.110.2.2 get_scancode()	546
5.110.2.3 to_string() [1/2]	546
5.110.2.4 to_string() [2/2]	546
5.111 input.hpp	547
5.112 liblava/frame/render_target.hpp File Reference	552
5.112.1 Detailed Description	553
5.112.2 Function Documentation	553
5.112.2.1 create_target()	553
5.112.2.2 create_target_no_triple_buffer()	554
5.112.2.3 create_target_v_sync()	554
5.113 render_target.hpp	554
5.114 liblava/frame/renderer.hpp File Reference	556

5.114.1 Detailed Description	556
5.115 <code>renderer.hpp</code>	557
5.116 <code>liblava/frame/swapchain.hpp</code> File Reference	557
5.116.1 Detailed Description	558
5.117 <code>swapchain.hpp</code>	558
5.118 <code>liblava/frame/window.hpp</code> File Reference	559
5.118.1 Detailed Description	560
5.118.2 Function Documentation	560
5.118.2.1 <code>create_surface()</code>	560
5.118.2.2 <code>get_window()</code>	560
5.119 <code>window.hpp</code>	561
5.120 <code>liblava/lava.hpp</code> File Reference	564
5.120.1 Detailed Description	564
5.121 <code>lava.hpp</code>	564
5.122 <code>liblava/resource.hpp</code> File Reference	565
5.122.1 Detailed Description	565
5.123 <code>resource.hpp</code>	565
5.124 <code>liblava/resource/buffer.hpp</code> File Reference	565
5.124.1 Detailed Description	566
5.124.2 Function Documentation	566
5.124.2.1 <code>buffer_usage_to_possible_access()</code>	566
5.124.2.2 <code>buffer_usage_to_possible_stages()</code>	566
5.125 <code>buffer.hpp</code>	567
5.126 <code>liblava/resource/format.hpp</code> File Reference	568
5.126.1 Detailed Description	569
5.126.2 Function Documentation	569
5.126.2.1 <code>find_supported_depth_format()</code>	569
5.126.2.2 <code>find_supported_format()</code>	570
5.126.2.3 <code>find_surface_format()</code>	570
5.126.2.4 <code>format_align_dim()</code>	570
5.126.2.5 <code>format_aspect_mask()</code>	571
5.126.2.6 <code>format_bgr()</code>	571
5.126.2.7 <code>format_block_dim()</code>	571
5.126.2.8 <code>format_block_size()</code> [1/2]	572
5.126.2.9 <code>format_block_size()</code> [2/2]	572
5.126.2.10 <code>format_depth()</code>	572
5.126.2.11 <code>format_depth_stencil()</code>	573
5.126.2.12 <code>format_num_blocks()</code>	573
5.126.2.13 <code>format_srgb()</code>	573
5.126.2.14 <code>format_stencil()</code>	574
5.126.2.15 <code>image_memory_barrier()</code>	574
5.126.2.16 <code>insert_image_memory_barrier()</code>	574

5.126.2.17 set_image_layout() [1/2]	575
5.126.2.18 set_image_layout() [2/2]	575
5.126.2.19 support_blit()	576
5.126.2.20 support_vertex_buffer_format()	576
5.127 format.hpp	577
5.128 liblava/resource/image.hpp File Reference	578
5.128.1 Detailed Description	578
5.128.2 Function Documentation	579
5.128.2.1 create_image()	579
5.128.2.2 grab_image()	579
5.129 image.hpp	579
5.130 liblava/resource/mesh.hpp File Reference	581
5.130.1 Detailed Description	582
5.130.2 Function Documentation	583
5.130.2.1 create_mesh()	583
5.130.2.2 create_mesh_data()	583
5.130.2.3 make_primitive_indices_cube()	584
5.130.2.4 make_primitive_normals_cube()	584
5.130.2.5 make_primitive_positions_cube()	584
5.130.2.6 make_primitive_uvs_cube()	585
5.131 mesh.hpp	585
5.132 liblava/resource/primitive.hpp File Reference	592
5.132.1 Detailed Description	593
5.133 primitive.hpp	593
5.134 liblava/resource/texture.hpp File Reference	594
5.134.1 Detailed Description	594
5.135 texture.hpp	595
5.136 liblava/test.hpp File Reference	596
5.136.1 Detailed Description	596
5.137 test.hpp	597
5.138 liblava/util.hpp File Reference	597
5.138.1 Detailed Description	597
5.139 util.hpp	597
5.140 liblava/util/hex.hpp File Reference	597
5.140.1 Detailed Description	600
5.140.2 Function Documentation	600
5.140.2.1 hex_calculate_inner_radius()	600
5.140.2.2 hex_cell_from_pair()	600
5.140.2.3 hex_corner_offset()	601
5.140.2.4 hex_diagonal()	601
5.140.2.5 hex_diagonal_neighbor()	601
5.140.2.6 hex_direction()	602



5.140.2.7 hex_distance()	602
5.140.2.8 hex_get()	602
5.140.2.9 hex_get_corner()	603
5.140.2.10 hex_get_s()	603
5.140.2.11 hex_is_valid()	603
5.140.2.12 hex_length()	604
5.140.2.13 hex_lerp()	604
5.140.2.14 hex_line()	604
5.140.2.15 hex_neighbor()	605
5.140.2.16 hex_opposite()	605
5.140.2.17 hex_pixel_to_cell()	605
5.140.2.18 hex_polygon_corners()	606
5.140.2.19 hex_q_doubled_from_cube()	606
5.140.2.20 hex_q_doubled_to_cube()	606
5.140.2.21 hex_q_offset_from_cube()	607
5.140.2.22 hex_q_offset_to_cube()	607
5.140.2.23 hex_r_doubled_from_cube()	607
5.140.2.24 hex_r_doubled_to_cube()	608
5.140.2.25 hex_r_offset_from_cube()	608
5.140.2.26 hex_r_offset_to_cube()	608
5.140.2.27 hex_round()	609
5.140.2.28 hex_to_pixel()	609
5.140.2.29 to_string()	609
5.140.3 Variable Documentation	610
5.140.3.1 hex_cardinal_directions	610
5.140.3.2 hex_diagonals	610
5.140.3.3 hex_directions	610
5.140.3.4 hex_layout_flat	611
5.140.3.5 hex_layout_point_y	611
5.141 hex.hpp	611
5.142 liblava/util/layer.hpp File Reference	616
5.142.1 Detailed Description	617
5.143 layer.hpp	617
5.144 liblava/util/log.hpp File Reference	618
5.144.1 Detailed Description	619
5.144.2 Function Documentation	619
5.144.2.1 logger()	619
5.144.2.2 sem_version_string()	620
5.144.2.3 semantic_version_string()	620
5.144.2.4 setup()	620
5.144.2.5 teardown()	620
5.144.2.6 to_string() [1/4]	620

5.144.2.7 to_string() [2/4]	621
5.144.2.8 to_string() [3/4]	621
5.144.2.9 to_string() [4/4]	621
5.144.2.10 version_string()	622
5.145 log.hpp	622
5.146 liblava/util/math.hpp File Reference	624
5.146.1 Detailed Description	625
5.146.2 Function Documentation	625
5.146.2.1 ceil_div()	625
5.146.2.2 hash256()	625
5.146.2.3 perspective_matrix()	625
5.146.3 Variable Documentation	626
5.146.3.1 default_color	626
5.147 math.hpp	626
5.148 liblava/util/random.hpp File Reference	628
5.148.1 Detailed Description	628
5.148.2 Function Documentation	628
5.148.2.1 random() [1/2]	628
5.148.2.2 random() [2/2]	628
5.149 random.hpp	629
5.150 liblava/util/telegram.hpp File Reference	630
5.150.1 Detailed Description	630
5.151 telegram.hpp	631
5.152 liblava/util/thread.hpp File Reference	633
5.152.1 Detailed Description	633
5.152.2 Function Documentation	633
5.152.2.1 sleep() [1/3]	633
5.152.2.2 sleep() [2/3]	634
5.152.2.3 sleep() [3/3]	634
5.153 thread.hpp	634

<b>Index</b>	<b>637</b>
--------------	------------

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

lava::app::about_info_setting . . . . .	15
lava::allocator . . . . .	16
lava::attachment . . . . .	27
lava::benchmark_data . . . . .	32
lava::descriptor::binding . . . . .	33
lava::c_data . . . . .	47
lava::json_file::callback . . . . .	49
lava::swapchain::callback . . . . .	49
lava::imgui::config . . . . .	64
lava::log::config . . . . .	65
lava::render_pipeline::create_info . . . . .	67
lava::device::create_param . . . . .	67
lava::instance::create_param . . . . .	70
lava::data . . . . .	71
lava::u_data . . . . .	325
lava::file_data . . . . .	115
lava::data_provider . . . . .	74
lava::instance::debug_config . . . . .	74
lava::device_table . . . . .	91
lava::device . . . . .	81
lava::driver . . . . .	102
lava::imgui::font . . . . .	124
lava::forward_shading . . . . .	125
lava::frame_env . . . . .	133
lava::gamepad . . . . .	134
lava::gamepad_manager . . . . .	137
lava::global_logger . . . . .	138
lava::hex_cell . . . . .	139
lava::hex_fractional_cell . . . . .	141
lava::hex_grid . . . . .	142
lava::hex_layout . . . . .	144
lava::hex_offset_coord . . . . .	144
lava::hex_orientation . . . . .	145
lava::hex_point . . . . .	145
lava::imgui::icon_font . . . . .	146

lava::id	147
lava::id_listeners< T >	148
lava::id_listeners< key_event >	148
lava::id_listeners< mouse_active_event >	148
lava::id_listeners< mouse_button_event >	148
lava::id_listeners< mouse_move_event >	148
lava::id_listeners< path_drop_event >	148
lava::id_listeners< scroll_event >	148
lava::id_registry< T, Meta >	150
lava::id_registry< lava::mesh_template, string >	150
lava::id_registry< lava::texture, string >	150
lava::ids	153
lava::image_data	162
lava::imgui	164
lava::input	170
lava::input_callback	172
lava::instance_info	177
lava::interface	177
lava::configurable	66
lava::app_config	25
lava::props	236
lava::entity	108
lava::block	36
lava::buffer	42
lava::camera	50
lava::command	56
lava::descriptor	75
lava::descriptor::pool	228
lava::device	81
lava::image	154
lava::layer	184
lava::mesh_template< T >	192
lava::physical_device	210
lava::pipeline	216
lava::compute_pipeline	59
lava::render_pipeline	258
lava::pipeline_layout	222
lava::render_pass	250
lava::render_target	272
lava::renderer	278
lava::subpass	293
lava::swapchain	306
lava::texture	315
lava::window	329
lava::frame	127
lava::app	18
lava::engine	104
lava::telegraph	314
lava::message_dispatcher	201
lava::props::item	178
lava::json_file	179
lava::key_event	182
lava::texture::layer	186
lava::layer_list	187
key_event::list	
lava::input_events< key_event >	173
mouse_active_event::list	
lava::input_events< mouse_active_event >	173

mouse_button_event::list	
lava::input_events< mouse_button_event > . . . . .	173
mouse_move_event::list	
lava::input_events< mouse_move_event > . . . . .	173
path_drop_event::list	
lava::input_events< path_drop_event > . . . . .	173
scroll_event::list	
lava::input_events< scroll_event > . . . . .	173
T::list	
lava::input_events< T > . . . . .	173
lava::mesh_meta . . . . .	192
lava::mesh_template_data< T > . . . . .	199
lava::mesh_template_data< vertex > . . . . .	199
lava::texture::mip_level . . . . .	203
lava::mouse_active_event . . . . .	204
lava::mouse_button_event . . . . .	205
lava::mouse_move_event . . . . .	206
lava::mouse_position . . . . .	207
lava::no_copy_no_move . . . . .	208
lava::entity . . . . .	108
lava::file . . . . .	110
lava::file_delete . . . . .	117
lava::file_system . . . . .	118
lava::frame . . . . .	127
lava::instance . . . . .	174
lava::memory . . . . .	190
lava::pair_hash . . . . .	209
lava::path_drop_event . . . . .	209
lava::platform . . . . .	226
lava::producer . . . . .	231
lava::pseudorandom_generator . . . . .	241
lava::queue . . . . .	242
lava::queue_family_info . . . . .	244
lava::queue_info . . . . .	245
lava::random_generator . . . . .	245
lava::rect . . . . .	247
lava::driver::result . . . . .	281
lava::reversion_wrapper< T > . . . . .	282
lava::run_time . . . . .	282
lava::scoped_label< T > . . . . .	283
lava::scroll_event . . . . .	284
lava::scroll_offset . . . . .	285
lava::semantic_version . . . . .	285
lava::pipeline::shader_stage . . . . .	286
lava::stage . . . . .	289
lava::staging . . . . .	290
lava::window::state . . . . .	292
lava::subpass_dependency . . . . .	300
lava::surface_format_request . . . . .	305
lava::target_callback . . . . .	312
lava::telegram . . . . .	312
lava::texture_file . . . . .	320
lava::thread_pool . . . . .	320
lava::timer . . . . .	321
lava::tooltip . . . . .	322
lava::tooltip_list . . . . .	323
lava::version . . . . .	327
lava::vertex . . . . .	327

lava::vk\_result . . . . . [328](#)

## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">lava::app::about_info_setting</a>	15
<a href="#">lava::allocator</a>	
Vulkan allocator	16
<a href="#">lava::app</a>	
Application with basic functionality	18
<a href="#">lava::app_config</a>	
Application configuration	25
<a href="#">lava::attachment</a>	
Attachment description	27
<a href="#">lava::benchmark_data</a>	
Benchmark data	32
<a href="#">lava::descriptor::binding</a>	
Descriptor binding	33
<a href="#">lava::block</a>	
Block of commands	36
<a href="#">lava::buffer</a>	
Buffer	42
<a href="#">lava::c_data</a>	
Const data wrapper	47
<a href="#">lava::json_file::callback</a>	
Json file callback	49
<a href="#">lava::swapchain::callback</a>	
Swapchain callback	49
<a href="#">lava::camera</a>	
First Person / Look At camera	50
<a href="#">lava::command</a>	
Block command	56
<a href="#">lava::compute_pipeline</a>	
Compute pipeline	59
<a href="#">lava::imgui::config</a>	
ImGui configuration	64
<a href="#">lava::log::config</a>	
Log configuration	65
<a href="#">lava::configurable</a>	
Configurable interface	66

<a href="#">lava::render_pipeline::create_info</a>	
Render pipeline create information . . . . .	67
<a href="#">lava::device::create_param</a>	
Device create parameters . . . . .	67
<a href="#">lava::instance::create_param</a>	
Instance create parameters . . . . .	70
<a href="#">lava::data</a>	
Data wrapper . . . . .	71
<a href="#">lava::data_provider</a>	
Data provider . . . . .	74
<a href="#">lava::instance::debug_config</a>	
Debug configuration . . . . .	74
<a href="#">lava::descriptor</a>	
Descriptor . . . . .	75
<a href="#">lava::device</a>	
Vulkan device . . . . .	81
<a href="#">lava::device_table</a>	
Device functions . . . . .	91
<a href="#">lava::driver</a>	
Stage driver . . . . .	102
<a href="#">lava::engine</a>	
Engine . . . . .	104
<a href="#">lava::entity</a>	
Entity . . . . .	108
<a href="#">lava::file</a>	
File . . . . .	110
<a href="#">lava::file_data</a>	
File data . . . . .	115
<a href="#">lava::file_delete</a>	
File delete guard . . . . .	117
<a href="#">lava::file_system</a>	
File system . . . . .	118
<a href="#">lava::imgui::font</a>	
ImGui font settings . . . . .	124
<a href="#">lava::forward_shading</a>	
Forward shading . . . . .	125
<a href="#">lava::frame</a>	
Framework . . . . .	127
<a href="#">lava::frame_env</a>	
Framework environment . . . . .	133
<a href="#">lava::gamepad</a>	
Gamepad . . . . .	134
<a href="#">lava::gamepad_manager</a>	
Gamepad manager . . . . .	137
<a href="#">lava::global_logger</a>	
Global logger . . . . .	138
<a href="#">lava::hex_cell</a>	
Hex cell . . . . .	139
<a href="#">lava::hex_fractional_cell</a>	
Hex fractional cell . . . . .	141
<a href="#">lava::hex_grid</a>	
Hex grid . . . . .	142
<a href="#">lava::hex_layout</a>	
Hex layout . . . . .	144
<a href="#">lava::hex_offset_coord</a>	
Hex offset coordinates . . . . .	144
<a href="#">lava::hex_orientation</a>	
Hex orientation . . . . .	145



<a href="#">lava::hex_point</a>	
Hex point	145
<a href="#">lava::imgui::icon_font</a>	
ImGui icon font settings	146
<a href="#">lava::id</a>	
Identification	147
<a href="#">lava::id_listeners&lt; T &gt;</a>	
Id listeners	148
<a href="#">lava::id_registry&lt; T, Meta &gt;</a>	
Id registry	150
<a href="#">lava::ids</a>	
Id factory	153
<a href="#">lava::image</a>	
Image	154
<a href="#">lava::image_data</a>	
Image data	162
<a href="#">lava::imgui</a>	
ImGui integration	164
<a href="#">lava::input</a>	
Input handling	170
<a href="#">lava::input_callback</a>	
Input callback	172
<a href="#">lava::input_events&lt; T &gt;</a>	
List of input events	173
<a href="#">lava::instance</a>	
Vulkan instance	174
<a href="#">lava::instance_info</a>	
Vulkan instance information	177
<a href="#">lava::interface</a>	
Interface	177
<a href="#">lava::props::item</a>	
Prop item	178
<a href="#">lava::json_file</a>	
Json file	179
<a href="#">lava::key_event</a>	
Key event	182
<a href="#">lava::layer</a>	
Layer	184
<a href="#">lava::texture::layer</a>	
Texture layer	186
<a href="#">lava::layer_list</a>	
Layer list	187
<a href="#">lava::memory</a>	
Vulkan memory	190
<a href="#">lava::mesh_meta</a>	
Mesh meta	192
<a href="#">lava::mesh_template&lt; T &gt;</a>	
Temporary templated mesh	192
<a href="#">lava::mesh_template_data&lt; T &gt;</a>	
Templated mesh data	199
<a href="#">lava::message_dispatcher</a>	
Message dispatcher	201
<a href="#">lava::texture::mip_level</a>	
Texture mip level	203
<a href="#">lava::mouse_active_event</a>	
Mouse active event	204
<a href="#">lava::mouse_button_event</a>	
Mouse button event	205

<a href="#">lava::mouse_move_event</a>	
Mouse move event	206
<a href="#">lava::mouse_position</a>	
Input mouse position	207
<a href="#">lava::no_copy_no_move</a>	
No copy and no move object	208
<a href="#">lava::pair_hash</a>	
Pair hash	209
<a href="#">lava::path_drop_event</a>	
Path drop event	209
<a href="#">lava::physical_device</a>	
Vulkan physical device	210
<a href="#">lava::pipeline</a>	
Pipeline	216
<a href="#">lava::pipeline_layout</a>	
Pipeline layout	222
<a href="#">lava::platform</a>	
Stage platform	226
<a href="#">lava::descriptor::pool</a>	
Descriptor pool	228
<a href="#">lava::producer</a>	
Producer	231
<a href="#">lava::props</a>	
Props	236
<a href="#">lava::pseudorandom_generator</a>	
Pseudorandom generator	241
<a href="#">lava::queue</a>	
Device queue	242
<a href="#">lava::queue_family_info</a>	
Queue family information	244
<a href="#">lava::queue_info</a>	
Queue information	245
<a href="#">lava::random_generator</a>	
Random generator	245
<a href="#">lava::rect</a>	
Rectangle	247
<a href="#">lava::render_pass</a>	
Render pass	250
<a href="#">lava::render_pipeline</a>	
Render pipeline (Graphics)	258
<a href="#">lava::render_target</a>	
Render target	272
<a href="#">lava::renderer</a>	
Plain renderer	278
<a href="#">lava::driver::result</a>	
Driver result	281
<a href="#">lava::reversion_wrapper&lt; T &gt;</a>	
Reversion Wrapper	282
<a href="#">lava::run_time</a>	
Run time	282
<a href="#">lava::scoped_label&lt; T &gt;</a>	
Scoped debug util label	283
<a href="#">lava::scroll_event</a>	
Scroll event	284
<a href="#">lava::scroll_offset</a>	
Input scroll offset	285
<a href="#">lava::semantic_version</a>	
Semantic version	285

<a href="#">lava::pipeline::shader_stage</a>	
Shader stage . . . . .	286
<a href="#">lava::stage</a>	
Stage . . . . .	289
<a href="#">lava::staging</a>	
Texture staging . . . . .	290
<a href="#">lava::window::state</a>	
Window state . . . . .	292
<a href="#">lava::subpass</a>	
Subpass . . . . .	293
<a href="#">lava::subpass_dependency</a>	
Subpass dependency . . . . .	300
<a href="#">lava::surface_format_request</a>	
Surface format request . . . . .	305
<a href="#">lava::swapchain</a>	
Swaphchain . . . . .	306
<a href="#">lava::target_callback</a>	
Target callback . . . . .	312
<a href="#">lava::telegram</a>	
Telegram . . . . .	312
<a href="#">lava::telegraph</a>	
Telegraph station . . . . .	314
<a href="#">lava::texture</a>	
Texture . . . . .	315
<a href="#">lava::texture_file</a>	
Texture file path with format . . . . .	320
<a href="#">lava::thread_pool</a>	
Thread pool . . . . .	320
<a href="#">lava::timer</a>	
Timer . . . . .	321
<a href="#">lava::tooltip</a>	
Tooltip . . . . .	322
<a href="#">lava::tooltip_list</a>	
Tooltip list . . . . .	323
<a href="#">lava::u_data</a>	
Unique data wrapper . . . . .	325
<a href="#">lava::version</a>	
Version . . . . .	327
<a href="#">lava::vertex</a>	
Vertex . . . . .	327
<a href="#">lava::vk_result</a>	
Vulkan result . . . . .	328
<a href="#">lava::window</a>	
Window . . . . .	329



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

liblava/ <a href="#">app.hpp</a>	
App module . . . . .	345
liblava/ <a href="#">asset.hpp</a>	
Asset module . . . . .	375
liblava/ <a href="#">base.hpp</a>	
Base module . . . . .	381
liblava/ <a href="#">block.hpp</a>	
Block module . . . . .	450
liblava/ <a href="#">core.hpp</a>	
Core module . . . . .	472
liblava/ <a href="#">engine.hpp</a>	
Engine module . . . . .	515
liblava/ <a href="#">file.hpp</a>	
File module . . . . .	520
liblava/ <a href="#">frame.hpp</a>	
Frame module . . . . .	530
liblava/ <a href="#">lava.hpp</a>	
All lava modules . . . . .	564
liblava/ <a href="#">resource.hpp</a>	
Resource module . . . . .	565
liblava/ <a href="#">test.hpp</a>	
Unit tests . . . . .	596
liblava/ <a href="#">util.hpp</a>	
Util module . . . . .	597
liblava/app/ <a href="#">app.hpp</a>	
Application with basic functionality . . . . .	346
liblava/app/ <a href="#">benchmark.hpp</a>	
Benchmark . . . . .	348
liblava/app/ <a href="#">camera.hpp</a>	
First Person / Look At camera . . . . .	351
liblava/app/ <a href="#">config.hpp</a>	
Application configuration . . . . .	354
liblava/app/ <a href="#">forward_shading.hpp</a>	
Forward shading . . . . .	355
liblava/app/ <a href="#">icon.hpp</a>	
App default icon data . . . . .	356

liblava/app/ <a href="#">imgui.hpp</a>	
ImGui integration	370
liblava/asset/ <a href="#">load_image.hpp</a>	
Load image data from file and memory	376
liblava/asset/ <a href="#">load_mesh.hpp</a>	
Load mesh from file	377
liblava/asset/ <a href="#">load_texture.hpp</a>	
Load texture from file	378
liblava/asset/ <a href="#">write_image.hpp</a>	
Write image data to file	380
liblava/base/ <a href="#">base.hpp</a>	
Vulkan base types	382
liblava/base/ <a href="#">debug_utils.hpp</a>	
Debug utilities	389
liblava/base/ <a href="#">device.hpp</a>	
Vulkan device	428
liblava/base/ <a href="#">device_table.hpp</a>	
Device functions	432
liblava/base/ <a href="#">instance.hpp</a>	
Vulkan instance	437
liblava/base/ <a href="#">memory.hpp</a>	
Vulkan allocator	440
liblava/base/ <a href="#">physical_device.hpp</a>	
Vulkan physical device	443
liblava/base/ <a href="#">platform.hpp</a>	
Stage platform	445
liblava/base/ <a href="#">queue.hpp</a>	
Device queue	446
liblava/block/ <a href="#">attachment.hpp</a>	
Attachment description	453
liblava/block/ <a href="#">block.hpp</a>	
Command buffer model	451
liblava/block/ <a href="#">compute_pipeline.hpp</a>	
Compute pipeline	454
liblava/block/ <a href="#">descriptor.hpp</a>	
Descriptor definition	456
liblava/block/ <a href="#">pipeline.hpp</a>	
Pipeline	458
liblava/block/ <a href="#">pipeline_layout.hpp</a>	
Pipeline layout	461
liblava/block/ <a href="#">render_pass.hpp</a>	
Render pass	463
liblava/block/ <a href="#">render_pipeline.hpp</a>	
Render pipeline (Graphics)	465
liblava/block/ <a href="#">subpass.hpp</a>	
Subpass	469
liblava/core/ <a href="#">data.hpp</a>	
Data wrapper	472
liblava/core/ <a href="#">id.hpp</a>	
Object Identification	478
liblava/core/ <a href="#">misc.hpp</a>	
Miscellaneous helpers	482
liblava/core/ <a href="#">time.hpp</a>	
Run time	492
liblava/core/ <a href="#">types.hpp</a>	
Basic types	499
liblava/core/ <a href="#">version.hpp</a>	
Version information	513

liblava/engine/engine.hpp	
Engine	516
liblava/engine/producer.hpp	
Producer	517
liblava/engine/props.hpp	
Props	519
liblava/file/file.hpp	
File access	521
liblava/file/file_system.hpp	
File system	523
liblava/file/file_utils.hpp	
File utilities	525
liblava/file/json.hpp	528
liblava/file/json_file.hpp	
Json file	529
liblava/frame/argh.hpp	
Json	535
liblava/frame/driver.hpp	
Stage driver	537
liblava/frame/frame.hpp	
Framework	531
liblava/frame/gamepad.hpp	
Gamepad manager	540
liblava/frame/input.hpp	
Input handling	543
liblava/frame/render_target.hpp	
Render target	552
liblava/frame/renderer.hpp	
Plain renderer	556
liblava/frame/swapchain.hpp	
Swapchain	557
liblava/frame/window.hpp	
Window	559
liblava/resource/buffer.hpp	
Vulkan buffer	565
liblava/resource/format.hpp	
Vulkan format	568
liblava/resource/image.hpp	
Vulkan image	578
liblava/resource/mesh.hpp	
Vulkan mesh	581
liblava/resource/primitive.hpp	
Vulkan primitives	592
liblava/resource/texture.hpp	
Vulkan texture	594
liblava/util/hex.hpp	
Hex point, cell and grid	597
liblava/util/layer.hpp	
Layering	616
liblava/util/log.hpp	
Logging	618
liblava/util/math.hpp	
Math helpers	624
liblava/util/random.hpp	
Random generator	628
liblava/util/telegram.hpp	
Message dispatcher	630

liblava/util/[thread.hpp](#)  
Thread pool . . . . . [633](#)



# Chapter 4

## Class Documentation

### 4.1 `lava::app::about_info_setting` Struct Reference

```
#include <app.hpp>
```

#### Static Public Member Functions

- static `about_info_setting all()`  
*Get about info setting for all.*

#### Public Attributes

- bool `draw_separator` = true  
*Draw with separator.*
- bool `draw_fps` = true  
*Draw with fps.*
- bool `draw_spacing` = true  
*Draw with spacing.*

#### 4.1.1 Detailed Description

About information setting

#### 4.1.2 Member Function Documentation

##### 4.1.2.1 `all()`

```
static about_info_setting lava::app::about_info_setting::all () [inline], [static]
```

Get about info setting for all.

#### Returns

`about_info_setting` About information setting

The documentation for this struct was generated from the following file:

- `liblava/app/app.hpp`

## 4.2 lava::allocator Struct Reference

Vulkan allocator.

```
#include <memory.hpp>
```

### Public Types

- using **s\_ptr** = std::shared\_ptr<allocator>  
*Shared pointer to a allocator.*
- using **device\_c\_ptr** = device const\*  
*Const pointer to device.*

### Public Member Functions

- **allocator** ()=default  
*Construct a new allocator.*
- **allocator** (VmaAllocator allocator)  
*Construct a new allocator.*
- bool **create** (device\_c\_ptr device, VmaAllocatorCreateFlags flags=0)  
*Create a new allocator.*
- void **destroy** ()  
*Destroy the allocator.*
- bool **valid** () const  
*Check if allocator is valid.*
- VmaAllocator **get** () const  
*Get the VMA allocator.*

### Static Public Member Functions

- static **s\_ptr** **make** ()  
*Make a new allocator.*

### 4.2.1 Detailed Description

Vulkan allocator.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 allocator()

```
lava::allocator::allocator (
    VmaAllocator allocator) [inline], [explicit]
```

Construct a new allocator.

## Parameters

<i>allocator</i>	VMA allocator
------------------	---------------

## 4.2.3 Member Function Documentation

### 4.2.3.1 create()

```
bool lava::allocator::create (  
    device_c_ptr device,  
    VmaAllocatorCreateFlags flags = 0)
```

Create a new allocator.

## Parameters

<i>device</i>	Vulkan device
<i>flags</i>	VMA allocator create flags

## Returns

Create was successful or failed

### 4.2.3.2 get()

```
VmaAllocator lava::allocator::get () const [inline]
```

Get the VMA allocator.

## Returns

VmaAllocator VMA allocator

### 4.2.3.3 make()

```
static s_ptr lava::allocator::make () [inline], [static]
```

Make a new allocator.

## Returns

s\_ptr Shared pointer to allocator

#### 4.2.3.4 valid()

```
bool lava::allocator::valid () const [inline]
```

Check if allocator is valid.

##### Returns

Allocator is valid or not

The documentation for this struct was generated from the following file:

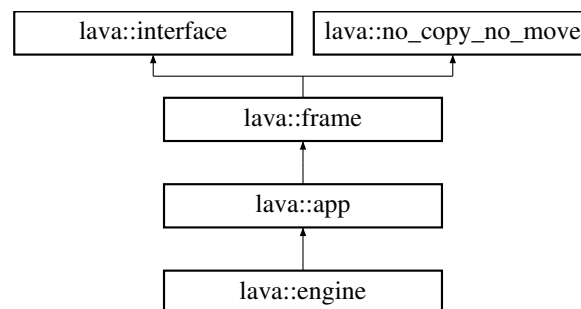
- liblava/base/[memory.hpp](#)

## 4.3 lava::app Struct Reference

Application with basic functionality.

```
#include <app.hpp>
```

Inheritance diagram for lava::app:



### Classes

- struct [about\\_info\\_setting](#)

### Public Types

- using **update\_func** = std::function<bool([delta](#))>  
*Update function.*
- using **create\_func** = std::function<bool()>  
*Create function.*
- using **destroy\_func** = std::function<void()>  
*Destroy function.*
- using **process\_func** = std::function<void(VkCommandBuffer, [index](#))>  
*Process function.*
- using **setup\_func** = std::function<bool()>  
*Set up function.*

## Public Types inherited from [lava::frame](#)

- using **s\_ptr** = std::shared\_ptr<[frame](#)>  
*Shared pointer to framework.*
- using **result** = [i32](#)  
*Framework result.*
- using **run\_func** = std::function<bool([id::ref](#))>  
*Run function.*
- using **run\_func\_ref** = [run\\_func](#) const&  
*Reference to run function.*
- using **run\_end\_func** = std::function<void()>  
*Run end function.*
- using **run\_end\_func\_ref** = [run\\_end\\_func](#) const&  
*Reference to run end function.*
- using **run\_once\_func** = std::function<bool()>  
*Run once function.*
- using **run\_once\_func\_ref** = [run\\_once\\_func](#) const&  
*Reference to run once function.*

## Public Member Functions

- [app](#) ([frame\\_env::ref](#) env)  
*Construct a new app.*
- [app](#) ([name](#) name, [argh::parser](#) cmd\_line={})  
*Construct a new app.*
- virtual bool [setup](#) ()  
*Set up the application.*
- bool [v\\_sync](#) () const  
*V-Sync setting.*
- bool [triple\\_buffer](#) () const  
*Triple buffering setting.*
- [ui32](#) [fps\\_cap](#) () const  
*Frames per second cap setting.*
- [ui32](#) [get\\_frame\\_counter](#) () const  
*Get the frame counter.*
- [string](#) [get\\_fps\\_info](#) () const  
*Get frames per second info.*
- void [draw\\_about](#) ([about\\_info\\_setting](#) setting=[about\\_info\\_setting::all](#)()) const  
*Draw about information.*
- [id::ref](#) [block\\_cmd](#) () const  
*Get id of the block command.*
- [string](#) [screenshot](#) ()  
*Take screenshot and save it to file.*
- void [switch\\_config](#) ([string\\_ref](#) config\_name)

## Public Member Functions inherited from [lava::frame](#)

- [frame](#) ([argh::parser cmd\\_line](#))  
*Construct a new framework.*
- [frame](#) ([frame\\_env env](#))  
*Construct a new framework.*
- [~frame](#) () override  
*Destroy the framework.*
- bool [ready](#) () const  
*Check if framework is ready.*
- [result run](#) ()  
*Run the framework.*
- bool [shut\\_down](#) ()  
*Shut down the framework.*
- [id add\\_run](#) ([run\\_func\\_ref func](#))  
*Add run to framework.*
- [id add\\_run\\_end](#) ([run\\_end\\_func\\_ref func](#))  
*Add run end to framework.*
- void [add\\_run\\_once](#) ([run\\_once\\_func\\_ref func](#))  
*Add run once to framework.*
- bool [remove](#) ([id::ref func\\_id](#))  
*Remove a function from framework.*
- [ms get\\_running\\_time](#) () const  
*Get the running time.*
- [r64 get\\_running\\_time\\_sec](#) () const  
*Get the running time in seconds.*
- [cmd\\_line get\\_cmd\\_line](#) () const  
*Get the command line arguments.*
- [frame\\_env::ref get\\_env](#) () const  
*Get the framework environment.*
- [name get\\_name](#) () const  
*Get the name of application.*
- bool [waiting\\_for\\_events](#) () const  
*Check if framework is waiting for events.*
- void [set\\_wait\\_for\\_events](#) (bool value=true)  
*Set wait for events in framework.*

## Public Member Functions inherited from [lava::interface](#)

- virtual [~interface](#) ()=default  
*Destroy the interface.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move const &](#))=delete  
*No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move const &](#))=delete  
*No move.*

## Public Attributes

- bool **headless** = false  
*Headless mode: no window, no input, no camera, no renderer, no block, no target, no shading, no gamepad. Enable it before calling the setup method.*
- [lava::window](#) **window**  
*Main window.*
- [lava::input](#) **input**  
*Window input.*
- [lava::imgui](#) **imgui**  
*ImGui handling.*
- [imgui::config](#) **imgui\_config**  
*ImGui configuration.*
- [tooltip\\_list](#) **tooltips**  
*Tooltip list.*
- [lava::device::ptr](#) **device** = nullptr  
*Vulkan device.*
- [lava::camera](#) **camera**  
*Main camera.*
- [gamepad](#) **pad**  
*Gamepad.*
- [lava::staging](#) **staging**  
*Texture staging.*
- [lava::block](#) **block**  
*Basic block.*
- [lava::renderer](#) **renderer**  
*Plain renderer.*
- [forward\\_shading](#) **shading**  
*Forward shading.*
- [render\\_target::s\\_ptr](#) **target**  
*Render target.*
- [file\\_system](#) **fs**  
*File system.*
- [VkPipelineCache](#) **pipeline\_cache** = nullptr  
*Pipeline cache.*
- [update\\_func](#) **on\_update**  
*Function called on application update.*
- [create\\_func](#) **on\_create**  
*Function called on application create.*
- [destroy\\_func](#) **on\_destroy**  
*Function called on application destroy.*
- [app\\_config](#) **config**  
*Application configuration.*
- [json\\_file](#) **config\_file**  
*Configuration file.*
- [process\\_func](#) **on\_process**  
*Function called on application process.*
- [setup\\_func](#) **on\_setup**  
*Function called on application setup.*

## Public Attributes inherited from [lava::frame](#)

- [lava::run\\_time](#) **run\_time**  
*Run time.*
- [lava::platform](#) **platform**  
*Stage platform.*
- [message\\_dispatcher](#) **telegraph**  
*Message dispatcher.*

### 4.3.1 Detailed Description

Application with basic functionality.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 `app()` [1/2]

```
lava::app::app (
    frame\_env::ref env) [explicit]
```

Construct a new app.

##### Parameters

<i>env</i>	Frame environment
------------	-------------------

#### 4.3.2.2 `app()` [2/2]

```
lava::app::app (
    name name,
    argh::parser cmd_line = {}) [explicit]
```

Construct a new app.

##### Parameters

<i>name</i>	Application name
<i>cmd_line</i>	Command line arguments

### 4.3.3 Member Function Documentation

#### 4.3.3.1 `block_cmd()`

```
id::ref lava::app::block_cmd () const [inline]
```

Get id of the block command.

##### Returns

[id::ref](#) Id to access the command

#### 4.3.3.2 `draw_about()`

```
void lava::app::draw_about (
    about\_info\_setting setting = about\_info\_setting::all\(\)) const
```

Draw about information.



## Parameters

<i>setting</i>	Setting
----------------	---------

**4.3.3.3 fps\_cap()**

```
ui32 lava::app::fps_cap () const [inline]
```

Frames per second cap setting.

## Returns

Frames per second cap value (deactivated: 0)

**4.3.3.4 get\_fps\_info()**

```
string lava::app::get_fps_info () const
```

Get frames per second info.

## Returns

string Frames per second info

**4.3.3.5 get\_frame\_counter()**

```
ui32 lava::app::get_frame_counter () const [inline]
```

Get the frame counter.

## Returns

ui32 Number of rendered frames

**4.3.3.6 screenshot()**

```
string lava::app::screenshot ()
```

Take screenshot and save it to file.

## Returns

string Screenshot file path (empty: failed)

#### 4.3.3.7 setup()

```
virtual bool lava::app::setup () [virtual]
```

Set up the application.

##### Returns

Setup was successful or failed

Reimplemented in [lava::engine](#).

#### 4.3.3.8 switch\_config()

```
void lava::app::switch_config (  
    string\_ref config_name)
```

Switch config name

## Parameters

<i>config_name</i>	Config name
--------------------	-------------

### 4.3.3.9 triple\_buffer()

```
bool lava::app::triple_buffer () const [inline]
```

Triple buffering setting.

## Returns

VK\_PRESENT\_MODE\_MAILBOX\_KHR preferred over VK\_PRESENT\_MODE\_IMMEDIATE\_KHR or not

### 4.3.3.10 v\_sync()

```
bool lava::app::v_sync () const [inline]
```

V-Sync setting.

## Returns

V-Sync is active or not

The documentation for this struct was generated from the following file:

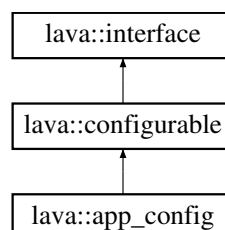
- [liblava/app/app.hpp](#)

## 4.4 lava::app\_config Struct Reference

Application configuration.

```
#include <config.hpp>
```

Inheritance diagram for lava::app\_config:



## Public Member Functions

- void [set\\_json](#) (json\_ref j) override
- json [get\\_json](#) () const override
- void **update\_window\_state** ()  
*Update window state.*

## Public Member Functions inherited from [lava::configurable](#)

## Public Member Functions inherited from [lava::interface](#)

- virtual  $\sim$ **interface** ()=default  
*Destroy the interface.*

## Public Attributes

- [app](#) \* **context** = nullptr  
*Application.*
- [name](#) **org** = [\\_liblava\\_](#)  
*Organization name.*
- [name](#) **ext** = "zip"  
*Preferred compression file format.*
- bool **save\_window** = true  
*Save window state.*
- bool **handle\_key\_events** = true  
*Handle key events.*
- bool **v\_sync** = false  
*Activate V-Sync.*
- bool **triple\_buffer** = true  
*Prefer VK\_PRESENT\_MODE\_MAILBOX\_KHR over VK\_PRESENT\_MODE\_IMMEDIATE\_KHR.*
- [ui32](#) **fps\_cap** = 0  
*Frames per second cap.*
- [surface\\_format\\_request](#) **surface**  
*Request surface formats.*
- [index](#) **physical\_device** = 0  
*Physical device index.*
- [imgui::font](#) **imgui\_font**  
*ImGui font settings.*
- [string](#) **name\_id** = [\\_default\\_](#)  
*Identification.*
- [window::state::optional](#) **window\_state**  
*Window state if available.*

### 4.4.1 Detailed Description

Application configuration.

## 4.4.2 Member Function Documentation

### 4.4.2.1 get\_json()

```
json lava::app_config::get_json () const [override], [virtual]
```

See also

[configurable::get\\_json](#)

Implements [lava::configurable](#).

### 4.4.2.2 set\_json()

```
void lava::app_config::set_json (
    json_ref j) [override], [virtual]
```

See also

[configurable::set\\_json](#)

Implements [lava::configurable](#).

The documentation for this struct was generated from the following file:

- liblava/app/[config.hpp](#)

## 4.5 lava::attachment Struct Reference

Attachment description.

```
#include <attachment.hpp>
```

### Public Types

- using **s\_ptr** = std::shared\_ptr<[attachment](#)>  
*Shared pointer to attachment.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of attachments.*

## Public Member Functions

- [attachment](#) (VkFormat format=VK\_FORMAT\_UNDEFINED, VkSampleCountFlagBits samples=VK\_SAMPLE\_COUNT\_1\_BIT)  
*Construct a new attachment.*
- VkAttachmentDescription const & [get\\_description](#) () const  
*Get the description.*
- void [set\\_format](#) (VkFormat format)  
*Set the format.*
- void [set\\_samples](#) (VkSampleCountFlagBits samples)  
*Set the samples.*
- void [set\\_op](#) (VkAttachmentLoadOp load\_op, VkAttachmentStoreOp store\_op)  
*Set the op.*
- void [set\\_load\\_op](#) (VkAttachmentLoadOp load\_op)  
*Set the load op.*
- void [set\\_store\\_op](#) (VkAttachmentStoreOp store\_op)  
*Set the store op.*
- void [set\\_stencil\\_op](#) (VkAttachmentLoadOp load\_op, VkAttachmentStoreOp store\_op)  
*Set the stencil op.*
- void [set\\_stencil\\_load\\_op](#) (VkAttachmentLoadOp load\_op)  
*Set the stencil load op.*
- void [set\\_stencil\\_store\\_op](#) (VkAttachmentStoreOp store\_op)  
*Set the stencil store op.*
- void [set\\_layouts](#) (VkImageLayout initial, VkImageLayout final)  
*Set the layouts.*
- void [set\\_initial\\_layout](#) (VkImageLayout layout)  
*Set the initial layout.*
- void [set\\_final\\_layout](#) (VkImageLayout layout)  
*Set the final layout.*

## Static Public Member Functions

- static [s\\_ptr make](#) (VkFormat format=VK\_FORMAT\_UNDEFINED, VkSampleCountFlagBits samples=VK\_SAMPLE\_COUNT\_1\_BIT)  
*Make a new attachment.*

## 4.5.1 Detailed Description

Attachment description.

## 4.5.2 Constructor & Destructor Documentation

### 4.5.2.1 attachment()

```
lava::attachment::attachment (
    VkFormat format = VK_FORMAT_UNDEFINED,
    VkSampleCountFlagBits samples = VK_SAMPLE_COUNT_1_BIT) [inline], [explicit]
```

Construct a new attachment.

## Parameters

<i>format</i>	Attachment format
<i>samples</i>	Sample count flag bits

## 4.5.3 Member Function Documentation

### 4.5.3.1 get\_description()

```
VkAttachmentDescription const & lava::attachment::get_description () const [inline]
```

Get the description.

## Returns

VkAttachmentDescription const& Attachment description

### 4.5.3.2 make()

```
static s_ptr lava::attachment::make (  
    VkFormat format = VK_FORMAT_UNDEFINED,  
    VkSampleCountFlagBits samples = VK_SAMPLE_COUNT_1_BIT) [inline], [static]
```

Make a new attachment.

## Parameters

<i>format</i>	Attachment format
<i>samples</i>	Sample count flag bits

## Returns

s\_ptr Shared pointer to attachment

### 4.5.3.3 set\_final\_layout()

```
void lava::attachment::set_final_layout (  
    VkImageLayout layout) [inline]
```

Set the final layout.

## Parameters

<i>layout</i>	Image layout
---------------	--------------

### 4.5.3.4 set\_format()

```
void lava::attachment::set_format (  
    VkFormat format) [inline]
```

Set the format.

## Parameters

<i>format</i>	Attachment format
---------------	-------------------

**4.5.3.5 set\_initial\_layout()**

```
void lava::attachment::set_initial_layout (
    VkImageLayout layout) [inline]
```

Set the initial layout.

## Parameters

<i>layout</i>	Image layout
---------------	--------------

**4.5.3.6 set\_layouts()**

```
void lava::attachment::set_layouts (
    VkImageLayout initial,
    VkImageLayout final) [inline]
```

Set the layouts.

## Parameters

<i>initial</i>	Initial image layout
<i>final</i>	Final image layout

**4.5.3.7 set\_load\_op()**

```
void lava::attachment::set_load_op (
    VkAttachmentLoadOp load_op) [inline]
```

Set the load op.

## Parameters

<i>load_op</i>	Attachment load op
----------------	--------------------

**4.5.3.8 set\_op()**

```
void lava::attachment::set_op (
    VkAttachmentLoadOp load_op,
    VkAttachmentStoreOp store_op) [inline]
```

Set the op.



## Parameters

<i>load_op</i>	Attachment load op
<i>store_op</i>	Attachment store op

**4.5.3.9 set\_samples()**

```
void lava::attachment::set_samples (
    VkSampleCountFlagBits samples) [inline]
```

Set the samples.

## Parameters

<i>samples</i>	Attachment sample count flag bits
----------------	-----------------------------------

**4.5.3.10 set\_stencil\_load\_op()**

```
void lava::attachment::set_stencil_load_op (
    VkAttachmentLoadOp load_op) [inline]
```

Set the stencil load op.

## Parameters

<i>load_op</i>	Attachment load op
----------------	--------------------

**4.5.3.11 set\_stencil\_op()**

```
void lava::attachment::set_stencil_op (
    VkAttachmentLoadOp load_op,
    VkAttachmentStoreOp store_op) [inline]
```

Set the stencil op.

## Parameters

<i>load_op</i>	Attachment load op
<i>store_op</i>	Attachment store op

**4.5.3.12 set\_stencil\_store\_op()**

```
void lava::attachment::set_stencil_store_op (
    VkAttachmentStoreOp store_op) [inline]
```

Set the stencil store op.

## Parameters

<code>store_op</code>	Attachment store op
-----------------------	---------------------

**4.5.3.13 set\_store\_op()**

```
void lava::attachment::set_store_op (
    VkAttachmentStoreOp store_op) [inline]
```

Set the store op.

## Parameters

<code>store_op</code>	Attachment store op
-----------------------	---------------------

The documentation for this struct was generated from the following file:

- liblava/block/[attachment.hpp](#)

**4.6 lava::benchmark\_data Struct Reference**

Benchmark data.

```
#include <benchmark.hpp>
```

**Public Types**

- using **list** = std::vector<[ui32](#)>  
*List of frame times.*

**Public Attributes**

- **ms time** = [ms](#){10000}  
*Benchmark duration.*
- **ms offset** = [ms](#){5000}  
*Warm up time.*
- **string file** = `_benchmark_json_`  
*Output file.*
- **string path**  
*Output path (empty: pref\_dir)*
- **bool exit** = true  
*Close app after benchmark.*
- **ui32 buffer\_size** = 100000  
*Pre-allocated buffer size for results.*
- **list values**  
*Benchmark results.*
- **index current** = 0  
*Current frame index.*
- **ms start\_timestamp** = [ms](#){0}  
*Benchmark start timestamp.*

### 4.6.1 Detailed Description

Benchmark data.

The documentation for this struct was generated from the following file:

- liblava/app/benchmark.hpp

## 4.7 lava::descriptor::binding Struct Reference

Descriptor binding.

```
#include <descriptor.hpp>
```

### Public Types

- using **s\_ptr** = std::shared\_ptr<binding>  
*Shared pointer to binding.*
- using **s\_list** = std::vector<s\_ptr>  
*List of bindings.*

### Public Member Functions

- **binding** ()  
*Construct a new binding.*
- VkDescriptorSetLayoutBinding const & **get** () const  
*Get the Vulkan descriptor set layout binding.*
- void **set** (index index)  
*Det the binding index.*
- void **set\_type** (VkDescriptorType descriptor\_type)  
*Set the type.*
- void **set\_count** (ui32 descriptor\_count)  
*Set the count.*
- void **set\_stage\_flags** (VkShaderStageFlags stage\_flags)  
*Set the stage flags.*
- void **set\_samplers** (VkSampler const \*immutable\_samplers)  
*Set the samplers.*

### Static Public Member Functions

- static **s\_ptr** **make** (index index)  
*Make a new descriptor binding.*

### 4.7.1 Detailed Description

Descriptor binding.

## 4.7.2 Member Function Documentation

### 4.7.2.1 get()

```
VkDescriptorSetLayoutBinding const & lava::descriptor::binding::get () const [inline]
```

Get the Vulkan descriptor set layout binding.

#### Returns

VkDescriptorSetLayoutBinding const& Vulkan binding

### 4.7.2.2 make()

```
static s_ptr lava::descriptor::binding::make (  
    index index) [inline], [static]
```

Make a new descriptor binding.

#### Parameters

<i>index</i>	Binding index
--------------	---------------

#### Returns

ptr Shared pointer to descriptor binding

### 4.7.2.3 set()

```
void lava::descriptor::binding::set (  
    index index) [inline]
```

Det the binding index.

#### Parameters

<i>index</i>	Binding index
--------------	---------------

### 4.7.2.4 set\_count()

```
void lava::descriptor::binding::set_count (  
    ui32 descriptor_count) [inline]
```

Set the count.

## Parameters

<i>descriptor_count</i>	Descriptor count
-------------------------	------------------

#### 4.7.2.5 set\_samplers()

```
void lava::descriptor::binding::set_samplers (
    VkSampler const * immutable_samplers) [inline]
```

Set the samplers.

## Parameters

<i>immutable_samplers</i>	Pointer to immutable samplers
---------------------------	-------------------------------

#### 4.7.2.6 set\_stage\_flags()

```
void lava::descriptor::binding::set_stage_flags (
    VkShaderStageFlags stage_flags) [inline]
```

Set the stage flags.

## Parameters

<i>stage_flags</i>	Shader stage flags
--------------------	--------------------

#### 4.7.2.7 set\_type()

```
void lava::descriptor::binding::set_type (
    VkDescriptorType descriptor_type) [inline]
```

Set the type.

## Parameters

<i>descriptor_type</i>	Descriptor type
------------------------	-----------------

The documentation for this struct was generated from the following file:

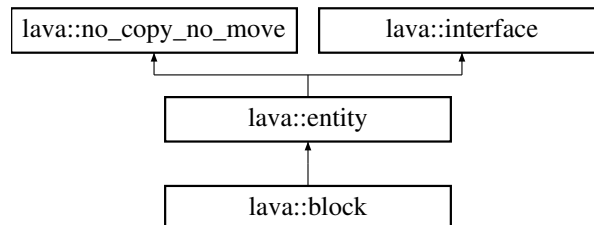
- liblava/block/[descriptor.hpp](#)

## 4.8 lava::block Struct Reference

Block of commands.

```
#include <block.hpp>
```

Inheritance diagram for lava::block:



### Public Types

- using **ptr** = [block\\*](#)  
*Pointer to block.*
- using **s\_ptr** = [std::shared\\_ptr<block>](#)  
*Shared pointer to block.*
- using **c\_ptr** = [block](#) const\*  
*Const pointer to block.*
- using **s\_map** = [std::map<id, s\\_ptr>](#)  
*Map of blocks.*
- using **c\_list** = [std::vector<c\\_ptr>](#)  
*List of blocks.*

### Public Member Functions

- **~block** ()  
*Destroy the block.*
- bool **create** ([device::ptr](#) device, [index](#) frame\_count, [index](#) queue\_family)  
*Create a new block.*
- void **destroy** ()  
*Destroy the block.*
- [index](#) **get\_frame\_count** () const  
*Get the frame count.*
- [id](#) **add\_cmd** ([command::process\\_func](#) func, bool active=true)
- [id](#) **add\_command** ([command::process\\_func](#) func, bool active=true)  
*Add a command.*
- void **remove\_cmd** ([id::ref](#) cmd\_id)
- void **remove\_command** ([id::ref](#) cmd\_id)  
*Remove the command.*
- bool **process** ([index](#) frame)  
*Process the block.*
- [index](#) **get\_current\_frame** () const  
*Get the current frame.*
- [VkCommandBuffer](#) **get\_command\_buffer** ([id::ref](#) cmd\_id) const

- Get the command buffer.*
- `VkCommandBuffer` `get_command_buffer` (`id::ref` cmd\_id, `index` frame) const  
*Get the command buffer.*
- `VkCommandBuffers` `collect_buffers` ()  
*Collect the buffers.*
- `command::s_map` const & `get_commands` () const  
*Get the commands.*
- `command::c_list` const & `get_cmd_order` () const  
*Get the cmd order.*
- `bool` `activated` (`id::ref` cmd\_id)  
*Check if command is activated.*
- `bool` `set_active` (`id::ref` cmd\_id, `bool` active=true)  
*Set the command active.*
- `device::ptr` `get_device` ()  
*Get the device.*

### Public Member Functions inherited from `lava::entity`

- `entity` ()  
*Construct a new entity.*
- `id::ref` `get_id` () const  
*Get the id of entity.*

### Public Member Functions inherited from `lava::no_copy_no_move`

- `no_copy_no_move` ()=default  
*Construct a new object.*
- `no_copy_no_move` (`no_copy_no_move` const &)=delete  
*No copy.*
- `void` `operator=` (`no_copy_no_move` const &)=delete  
*No move.*

### Public Member Functions inherited from `lava::interface`

- `virtual` `~interface` ()=default  
*Destroy the interface.*

### Static Public Member Functions

- `static` `s_ptr` `make` ()  
*Make a new block.*

## 4.8.1 Detailed Description

Block of commands.

## 4.8.2 Member Function Documentation

### 4.8.2.1 `activated()`

```
bool lava::block::activated (
    id::ref cmd_id)
```

Check if command is activated.

## Parameters

<i>cmd</i> ↔ <i>_id</i>	Command id
----------------------------	------------

## Returns

Command is active or not

**4.8.2.2 add\_cmd()**

```
id lava::block::add_cmd (
    command::process_func func,
    bool active = true)
```

## See also

[add\\_command](#)

**4.8.2.3 add\_command()**

```
id lava::block::add_command (
    command::process_func func,
    bool active = true) [inline]
```

Add a command.

## Parameters

<i>func</i>	Command function
<i>active</i>	Active state

## Returns

id Command id

**4.8.2.4 collect\_buffers()**

```
VkCommandBuffers lava::block::collect_buffers () [inline]
```

Collect the buffers.

## Returns

VkCommandBuffers List of Vulkan command buffers

**4.8.2.5 create()**

```
bool lava::block::create (
    device::ptr device,
    index frame_count,
    index queue_family)
```

Create a new block.



## Parameters

<i>device</i>	Vulkan device
<i>frame_count</i>	Number of frames
<i>queue_family</i>	Queue family index

## Returns

Create was successful or failed

## 4.8.2.6 get\_cmd\_order()

```
command::c_list const & lava::block::get_cmd_order () const [inline]
```

Get the cmd order.

## Returns

[command::c\\_list](#) const& List of commands

## 4.8.2.7 get\_command\_buffer() [1/2]

```
VkCommandBuffer lava::block::get_command_buffer (
    id::ref cmd_id) const [inline]
```

Get the command buffer.

## Parameters

<i>cmd</i> ↔ <i>_id</i>	Command id
----------------------------	------------

## Returns

VkCommandBuffer Vulkan command buffer

## 4.8.2.8 get\_command\_buffer() [2/2]

```
VkCommandBuffer lava::block::get_command_buffer (
    id::ref cmd_id,
    index frame) const [inline]
```

Get the command buffer.

## Parameters

<i>cmd</i> ↔ <i>_id</i>	Command id
<i>frame</i>	Frame index

## Returns

VkCommandBuffer Vulkan command buffer

#### 4.8.2.9 get\_commands()

```
command::s_map const & lava::block::get_commands () const [inline]
```

Get the commands.

Returns

`command::s_map` const& Map of commands

#### 4.8.2.10 get\_current\_frame()

```
index lava::block::get_current_frame () const [inline]
```

Get the current frame.

Returns

`index` Current frame

#### 4.8.2.11 get\_device()

```
device::ptr lava::block::get_device () [inline]
```

Get the device.

Returns

`device::ptr` Vulkan device

#### 4.8.2.12 get\_frame\_count()

```
index lava::block::get_frame_count () const [inline]
```

Get the frame count.

Returns

`index` Number of frames

#### 4.8.2.13 make()

```
static s_ptr lava::block::make () [inline], [static]
```

Make a new block.

Returns

`s_ptr` Shared pointer to block

#### 4.8.2.14 process()

```
bool lava::block::process (  
    index frame)
```

Process the block.

## Parameters

<i>frame</i>	Frame index
--------------	-------------

## Returns

Process was successful or aborted

**4.8.2.15 remove\_cmd()**

```
void lava::block::remove_cmd (  
    id::ref cmd_id)
```

## See also

[remove\\_command](#)

**4.8.2.16 remove\_command()**

```
void lava::block::remove_command (  
    id::ref cmd_id) [inline]
```

Remove the command.

## Parameters

<i>cmd</i> ↔ <i>_id</i>	Command id
----------------------------	------------

**4.8.2.17 set\_active()**

```
bool lava::block::set_active (  
    id::ref cmd_id,  
    bool active = true)
```

Set the command active.

## Parameters

<i>cmd</i> ↔ <i>_id</i>	Command id
<i>active</i>	Active state

## Returns

Set was successful or failed

The documentation for this struct was generated from the following file:

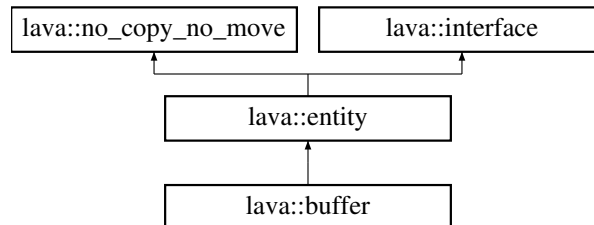
- [liblava/block/block.hpp](#)

## 4.9 lava::buffer Struct Reference

Buffer.

```
#include <buffer.hpp>
```

Inheritance diagram for lava::buffer:



### Public Types

- using **s\_ptr** = std::shared\_ptr<buffer>  
*Shared pointer to buffer.*
- using **s\_list** = std::vector<s\_ptr>  
*List of buffers.*

### Public Member Functions

- **~buffer** ()  
*Destroy the buffer.*
- bool **create** (device::ptr device, void const \*data, size\_t size, VkBufferUsageFlags usage, bool mapped=false, VmaMemoryUsage memory\_usage=VMA\_MEMORY\_USAGE\_GPU\_ONLY, VkSharingMode sharing\_mode=VK\_SHARING\_MODE\_EXCLUSIVE, std::vector< ui32 > const &shared\_queue\_family\_indices={}, i32 alignment=undef)  
*Create a new buffer.*
- bool **create\_mapped** (device::ptr device, void const \*data, size\_t size, VkBufferUsageFlags usage, VmaMemoryUsage memory\_usage=VMA\_MEMORY\_USAGE\_CPU\_TO\_GPU, VkSharingMode sharing\_mode=VK\_SHARING\_MODE\_EXCLUSIVE, std::vector< ui32 > const &shared\_queue\_family\_indices={}, i32 alignment=undef)  
*Create a new mapped buffer.*
- void **destroy** ()  
*Destroy the buffer.*
- device::ptr **get\_device** ()  
*Get the device.*
- bool **valid** () const  
*Check if the buffer is valid.*
- VkBuffer **get** () const  
*Get the buffer.*
- VkDescriptorBufferInfo const \* **get\_descriptor\_info** () const  
*Get the descriptor information.*
- VkDeviceAddress **get\_address** () const  
*Get the address of the buffer.*
- VkDeviceSize **get\_size** () const

- Get the size of the buffer.*
- void \* [get\\_mapped\\_data](#) () const  
*Get the mapped data.*
- VkDeviceMemory [get\\_device\\_memory](#) () const  
*Get the device memory of the buffer.*
- void [flush](#) (VkDeviceSize offset=0, VkDeviceSize size=VK\_WHOLE\_SIZE)  
*Flush the buffer data.*
- VmaAllocation const & [get\\_allocation](#) () const  
*Get the allocation.*
- VmaAllocationInfo const & [get\\_allocation\\_info](#) () const  
*Get the allocation information.*

### Public Member Functions inherited from [lava::entity](#)

- [entity](#) ()  
*Construct a new entity.*
- [id::ref get\\_id](#) () const  
*Get the id of entity.*

### Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

### Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

### Static Public Member Functions

- static [s\\_ptr make](#) ()  
*Make a new buffer.*

## 4.9.1 Detailed Description

Buffer.

## 4.9.2 Member Function Documentation

### 4.9.2.1 create()

```
bool lava::buffer::create (
    device::ptr device,
    void const * data,
    size_t size,
    VkBufferUsageFlags usage,
    bool mapped = false,
    VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_GPU_ONLY,
    VkSharingMode sharing_mode = VK_SHARING_MODE_EXCLUSIVE,
    std::vector< ui32 > const & shared_queue_family_indices = {},
    i32 alignment = undef)
```

Create a new buffer.

#### Parameters

<i>device</i>	Vulkan device
<i>data</i>	Buffer data
<i>size</i>	Data size
<i>usage</i>	Buffer usage flags
<i>mapped</i>	Map the buffer
<i>memory_usage</i>	Memory usage
<i>sharing_mode</i>	Sharing mode
<i>shared_queue_family_indices</i>	Queue indices (ignored unless <i>sharing_mode</i> == VK_SHARING_MODE_CONCURRENT)
<i>alignment</i>	Minimum alignment to be used when placing the buffer inside a larger memory block negative -> no minimum alignment

#### Returns

Create was successful or failed

### 4.9.2.2 create\_mapped()

```
bool lava::buffer::create_mapped (
    device::ptr device,
    void const * data,
    size_t size,
    VkBufferUsageFlags usage,
    VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_CPU_TO_GPU,
    VkSharingMode sharing_mode = VK_SHARING_MODE_EXCLUSIVE,
    std::vector< ui32 > const & shared_queue_family_indices = {},
    i32 alignment = undef)
```

Create a new mapped buffer.

#### Parameters

<i>device</i>	Vulkan device
<i>data</i>	Buffer data

## Parameters

<i>size</i>	Data size
<i>usage</i>	Buffer usage flags
<i>memory_usage</i>	Memory usage
<i>sharing_mode</i>	Sharing mode
<i>shared_queue_family_indices</i>	Queue indices (ignored unless <i>sharing_mode</i> == VK_SHARING_MODE_CONCURRENT)
<i>alignment</i>	Minimum alignment to be used when placing the buffer inside a larger memory block negative -> no minimum alignment

## Returns

Create was successful or failed

## 4.9.2.3 flush()

```
void lava::buffer::flush (  
    VkDeviceSize offset = 0,  
    VkDeviceSize size = VK_WHOLE_SIZE)
```

Flush the buffer data.

## Parameters

<i>offset</i>	Offset device size
<i>size</i>	Data device size

## 4.9.2.4 get()

```
VkBuffer lava::buffer::get () const [inline]
```

Get the buffer.

## Returns

VkBuffer Vulkan buffer

## 4.9.2.5 get\_address()

```
VkDeviceAddress lava::buffer::get_address () const
```

Get the address of the buffer.

## Returns

VkDeviceAddress Device address

#### 4.9.2.6 get\_allocation()

```
VmaAllocation const & lava::buffer::get_allocation () const [inline]
```

Get the allocation.

##### Returns

VmaAllocation const& Allocation

#### 4.9.2.7 get\_allocation\_info()

```
VmaAllocationInfo const & lava::buffer::get_allocation_info () const [inline]
```

Get the allocation information.

##### Returns

VmaAllocationInfo const& Allocation information

#### 4.9.2.8 get\_descriptor\_info()

```
VkDescriptorBufferInfo const * lava::buffer::get_descriptor_info () const [inline]
```

Get the descriptor information.

##### Returns

VkDescriptorBufferInfo const\* Descriptor buffer information

#### 4.9.2.9 get\_device()

```
device::ptr lava::buffer::get_device () [inline]
```

Get the device.

##### Returns

device::ptr Vulkan device

#### 4.9.2.10 get\_device\_memory()

```
VkDeviceMemory lava::buffer::get_device_memory () const [inline]
```

Get the device memory of the buffer.

##### Returns

VkDeviceMemory Device memory



#### 4.9.2.11 get\_mapped\_data()

```
void * lava::buffer::get_mapped_data () const [inline]
```

Get the mapped data.

##### Returns

void\* Pointer to data

#### 4.9.2.12 get\_size()

```
VkDeviceSize lava::buffer::get_size () const [inline]
```

Get the size of the buffer.

##### Returns

VkDeviceSize Device size

#### 4.9.2.13 make()

```
static s_ptr lava::buffer::make () [inline], [static]
```

Make a new buffer.

##### Returns

s\_ptr Shared pointer to buffer

#### 4.9.2.14 valid()

```
bool lava::buffer::valid () const [inline]
```

Check if the buffer is valid.

##### Returns

Buffer is valid or not

The documentation for this struct was generated from the following file:

- [liblava/resource/buffer.hpp](#)

## 4.10 lava::c\_data Struct Reference

Const data wrapper.

```
#include <data.hpp>
```

## Public Types

- using **ref** = [c\\_data](#) const&  
*Reference to const data wrapper.*

## Public Member Functions

- **c\_data** ()=default  
*Construct a new const data.*
- **c\_data** (void const \*[addr](#), [size\\_t](#) length)  
*Construct a new const data.*
- **c\_data** ([data::ref](#) data)  
*Construct a new const data from other data.*

## Public Attributes

- [data::c\\_ptr](#) **addr** = nullptr  
*Const data pointer.*
- [size\\_t](#) **size** = 0  
*Size of data.*

### 4.10.1 Detailed Description

Const data wrapper.

### 4.10.2 Constructor & Destructor Documentation

#### 4.10.2.1 **c\_data()** [1/2]

```
lava::c_data::c_data (
    void const * addr,
    size\_t length) [inline]
```

Construct a new const data.

#### Parameters

<i>addr</i>	Pointer to data
<i>length</i>	Length of data

#### 4.10.2.2 **c\_data()** [2/2]

```
lava::c_data::c_data (
    data::ref data) [inline]
```

Construct a new const data from other data.

## Parameters

<code>data</code>	Source data
-------------------	-------------

The documentation for this struct was generated from the following file:

- [liblava/core/data.hpp](#)

## 4.11 `lava::json_file::callback` Struct Reference

Json file callback.

```
#include <json_file.hpp>
```

### Public Types

- using **list** = `std::vector<callback\*>`  
*List of callbacks.*
- using **load\_func** = `std::function<void(json_ref)>`  
*Load function.*
- using **save\_func** = `std::function<json()>`  
*Save function.*

### Public Attributes

- [load\\_func](#) **on\_load**  
*Called on load.*
- [save\\_func](#) **on\_save**  
*Called on save.*

### 4.11.1 Detailed Description

Json file callback.

The documentation for this struct was generated from the following file:

- [liblava/file/json\\_file.hpp](#)

## 4.12 `lava::swapchain::callback` Struct Reference

Swapchain callback.

```
#include <swapchain.hpp>
```

## Public Types

- using **list** = std::vector<callback\*>  
*List of callbacks.*
- using **created\_func** = std::function<bool()>  
*Created function.*
- using **destroyed\_func** = std::function<void()>  
*Destroyed function.*

## Public Attributes

- **created\_func** on\_created  
*Called on swapchain created.*
- **destroyed\_func** on\_destroyed  
*Called on swapchain destroyed.*

### 4.12.1 Detailed Description

Swapchain callback.

The documentation for this struct was generated from the following file:

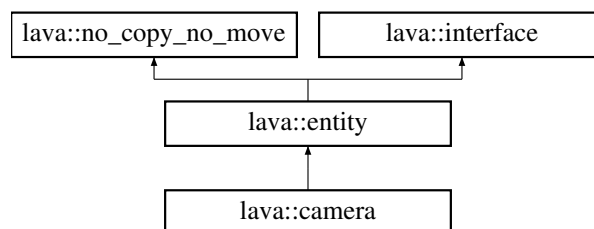
- liblava/frame/swapchain.hpp

## 4.13 lava::camera Struct Reference

First Person / Look At camera.

```
#include <camera.hpp>
```

Inheritance diagram for lava::camera:



## Public Types

- enum class **mode** : index { **first\_person** = 0 , **look\_at** }  
*Camera modes.*
- using **ptr** = camera\*  
*Pointer to camera.*
- using **s\_ptr** = std::shared\_ptr<camera>  
*Shared pointer to camera.*
- using **s\_map** = std::map<id, s\_ptr>  
*Map of cameras.*
- using **s\_list** = std::vector<s\_ptr>  
*List of cameras.*

## Public Member Functions

- bool [create](#) (device::ptr device)  
*Create a camera.*
- void **destroy** ()  
*Destroy the camera.*
- void **update\_projection** ()  
*Update the projection.*
- void [update\\_view](#) (delta dt, [mouse\\_position](#) mouse\_pos)  
*Update the view with mouse position.*
- void [update\\_view](#) (delta dt, [gamepad::ref](#) pad)  
*Update the view with gamepad.*
- [mat4 get\\_view](#) () const  
*Get the camera's 4x4 view matrix.*
- [mat4 get\\_projection](#) () const  
*Get the camera's 4x4 projection matrix.*
- [mat4 calc\\_view\\_projection](#) () const  
*Calc the camera's combined 4x4 view/projection matrix.*
- bool [handle](#) ([key\\_event::ref](#) event)  
*Handle key event.*
- bool [handle](#) ([mouse\\_button\\_event::ref](#) event, [mouse\\_position](#) mouse\_pos)  
*Handle mouse button event.*
- bool [handle](#) ([scroll\\_event::ref](#) event)  
*Handle scroll event.*
- bool [valid](#) () const  
*Check if camera is valid.*
- VkDescriptorBufferInfo const \* [get\\_descriptor\\_info](#) () const  
*Get the descriptor buffer info.*
- void **upload** ()  
*Upload camera state.*
- void **stop** ()  
*Stop camera movement.*
- void **reset** ()  
*Reset camera.*
- void [set\\_active](#) (bool value=true)  
*Set camera active.*
- bool [activated](#) () const  
*Check if camera is activated.*
- bool [moving](#) () const  
*Check if camera is moving.*
- void [set\\_movement\\_keys](#) ([keys\\_ref](#) up, [keys\\_ref](#) down, [keys\\_ref](#) left, [keys\\_ref](#) right)  
*Set keys for moving this camera.*

## Public Member Functions inherited from [lava::entity](#)

- **entity** ()  
*Construct a new entity.*
- [id::ref get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Public Attributes

- **v3 position** = [v3](#)(0.f)  
*Camera position.*
- **v3 rotation** = [v3](#)(0.f)  
*Camera rotation.*
- **r32 rotation\_speed** = 20.f  
*Camera rotation speed.*
- **r32 movement\_speed** = 1.f  
*Camera movement speed.*
- **r32 zoom\_speed** = 20.f  
*Camera zoom speed.*
- **r32 fov** = 60.f  
*Field of view.*
- **r32 z\_near** = 0.1f  
*Distance to near clipping plane along the -Z axis.*
- **r32 z\_far** = 256.f  
*Distance to far clipping plane along the -Z axis.*
- **r32 aspect\_ratio** = 1.77f  
*Camera aspect ratio.*
- **mode mode** = mode::first\_person  
*Camera mode.*
- bool **lock\_z** = false  
*Lock Z axis movement.*
- bool **lock\_rotation** = false  
*Lock camera rotation.*

### 4.13.1 Detailed Description

First Person / Look At camera.

## 4.13.2 Member Function Documentation

### 4.13.2.1 activated()

```
bool lava::camera::activated () const [inline]
```

Check if camera is activated.

#### Returns

Camera is active or not

### 4.13.2.2 calc\_view\_projection()

```
mat4 lava::camera::calc_view_projection () const
```

Calc the camera's combined 4x4 view/projection matrix.

#### Returns

mat4 Combined view/projection matrix

### 4.13.2.3 create()

```
bool lava::camera::create (  
    device::ptr device)
```

Create a camera.

#### Parameters

<i>device</i>	Vulkan device
---------------	---------------

#### Returns

Create was successful or failed

### 4.13.2.4 get\_descriptor\_info()

```
VkDescriptorBufferInfo const * lava::camera::get_descriptor_info () const [inline]
```

Get the descriptor buffer info.

#### Returns

VkDescriptorBufferInfo const\* Descriptor buffer info

#### 4.13.2.5 get\_projection()

```
mat4 lava::camera::get_projection () const
```

Get the camera's 4x4 projection matrix.

##### Returns

mat4 Projection matrix

#### 4.13.2.6 get\_view()

```
mat4 lava::camera::get_view () const
```

Get the camera's 4x4 view matrix.

##### Returns

mat4 View matrix

#### 4.13.2.7 handle() [1/3]

```
bool lava::camera::handle (  
    key_event::ref event)
```

Handle key event.

##### Parameters

<i>event</i>	Key event
--------------	-----------

##### Returns

Event was handled or ignored

#### 4.13.2.8 handle() [2/3]

```
bool lava::camera::handle (  
    mouse_button_event::ref event,  
    mouse_position mouse_pos)
```

Handle mouse button event.

##### Parameters

<i>event</i>	Mouse button event
<i>mouse_pos</i>	Mouse position

##### Returns

Event was handled or ignored

#### 4.13.2.9 handle() [3/3]

```
bool lava::camera::handle (  
    scroll_event::ref event)
```

Handle scroll event.



## Parameters

<i>event</i>	Scroll event
--------------	--------------

## Returns

Event was handled or ignored

**4.13.2.10 moving()**

```
bool lava::camera::moving () const [inline]
```

Check if camera is moving.

## Returns

Camera is moving or does not move

**4.13.2.11 set\_active()**

```
void lava::camera::set_active (
    bool value = true) [inline]
```

Set camera active.

## Parameters

<i>value</i>	Active state
--------------	--------------

**4.13.2.12 set\_movement\_keys()**

```
void lava::camera::set_movement_keys (
    keys_ref up,
    keys_ref down,
    keys_ref left,
    keys_ref right) [inline]
```

Set keys for moving this camera.

## Parameters

<i>up</i>	Up inputs
<i>down</i>	Down inputs
<i>left</i>	Left inputs
<i>right</i>	Right inputs

**4.13.2.13 update\_view() [1/2]**

```
void lava::camera::update_view (
    delta dt,
    gamepad::ref pad)
```

Update the view with gamepad.

## Parameters

<i>dt</i>	Delta time
<i>pad</i>	Gamepad

**4.13.2.14 update\_view()** [2/2]

```
void lava::camera::update_view (
    delta dt,
    mouse_position mouse_pos)
```

Update the view with mouse position.

## Parameters

<i>dt</i>	Delta time
<i>mouse_pos</i>	Mouse position

**4.13.2.15 valid()**

```
bool lava::camera::valid () const [inline]
```

Check if camera is valid.

## Returns

Camera is valid or not

The documentation for this struct was generated from the following file:

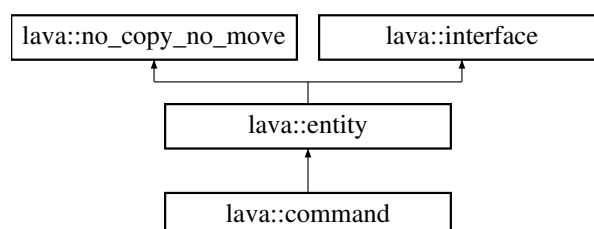
- liblava/app/[camera.hpp](#)

**4.14 lava::command Struct Reference**

Block command.

```
#include <block.hpp>
```

Inheritance diagram for lava::command:



## Public Types

- using **s\_ptr** = std::shared\_ptr<command>  
*Shared pointer to command.*
- using **c\_ptr** = command const\*  
*Const pointer to command.*
- using **s\_map** = std::map<id, s\_ptr>  
*Map of commands.*
- using **c\_list** = std::vector<c\_ptr>  
*List of commands.*
- using **process\_func** = std::function<void(VkCommandBuffer)>  
*Command process function.*

## Public Member Functions

- bool **create** (device::ptr device, index frame\_count, VkCommandPools command\_pools)  
*Create a new command.*
- void **destroy** (device::ptr device, VkCommandPools command\_pools)  
*Destroy the command.*

## Public Member Functions inherited from lava::entity

- **entity** ()  
*Construct a new entity.*
- id::ref **get\_id** () const  
*Get the id of entity.*

## Public Member Functions inherited from lava::no\_copy\_no\_move

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Member Functions inherited from lava::interface

- virtual ~**interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static s\_ptr **make** ()  
*Make a new command.*

## Public Attributes

- `VkCommandBuffers buffers = {}`  
*List of command buffers.*
- `process_func on_process`  
*Called on command process.*
- `bool active = true`  
*Active state.*

### 4.14.1 Detailed Description

Block command.

### 4.14.2 Member Function Documentation

#### 4.14.2.1 create()

```
bool lava::command::create (  
    device::ptr device,  
    index frame_count,  
    VkCommandPools command_pools)
```

Create a new command.

#### Parameters

<i>device</i>	Vulkan device
<i>frame_count</i>	Number of frames
<i>command_pools</i>	List of command pools

#### Returns

Create was successful or failed

#### 4.14.2.2 destroy()

```
void lava::command::destroy (  
    device::ptr device,  
    VkCommandPools command_pools)
```

Destroy the command.

#### Parameters

<i>device</i>	Vulkan device
<i>command_pools</i>	List of command pools

## 4.14.2.3 make()

```
static s_ptr lava::command::make () [inline], [static]
```

Make a new command.

## Returns

s\_ptr Shared pointer to command

The documentation for this struct was generated from the following file:

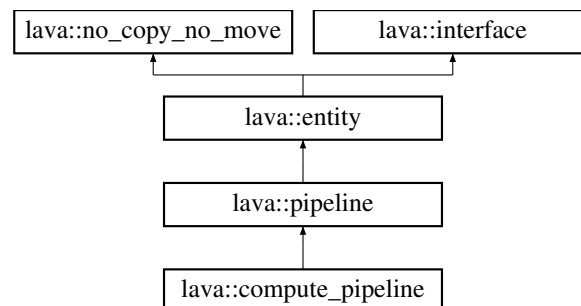
- liblava/block/[block.hpp](#)

## 4.15 lava::compute\_pipeline Struct Reference

Compute pipeline.

```
#include <compute_pipeline.hpp>
```

Inheritance diagram for lava::compute\_pipeline:



## Public Types

- using **s\_ptr** = std::shared\_ptr<[compute\\_pipeline](#)>  
*Shared pointer to compute pipeline.*
- using **s\_map** = std::map<[id](#), [s\\_ptr](#)>  
*Map of compute pipelines.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of compute pipelines.*

Public Types inherited from [lava::pipeline](#)

- using **s\_ptr** = std::shared\_ptr<[pipeline](#)>  
*Shared pointer to pipeline.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of pipelines.*
- using **process\_func** = std::function<void(VkCommandBuffer)>  
*Pipeline process function.*

## Public Member Functions

- void **bind** (VkCommandBuffer cmdBuffer) override  
*Bind the pipeline.*
- bool **set\_shader\_stage** (c\_data::ref data, VkShaderStageFlagBits stage)  
*Set shader stage.*
- void **set** (shader\_stage::s\_ptr const &stage)  
*Set shader stage.*
- shader\_stage::s\_ptr const & **get\_shader\_stage** () const  
*Get the shader stage.*
- void **copy\_to** (compute\_pipeline \*target) const  
*Copy configuration to target pipeline.*
- void **copy\_from** (s\_ptr const &source)  
*Copy configuration from source.*
- **pipeline** (device::ptr device, VkPipelineCache pipeline\_cache=0)  
*Pipeline constructors.*

## Public Member Functions inherited from [lava::pipeline](#)

- **pipeline** (device::ptr device, VkPipelineCache pipeline\_cache=0)  
*Construct a new pipeline.*
- **~pipeline** () override  
*Destroy the pipeline.*
- bool **create** ()  
*Create a new pipeline.*
- void **destroy** ()  
*Destroy the pipeline.*
- void **set\_active** (bool value=true)  
*Set pipeline active.*
- bool **activated** () const  
*Check if pipeline is active.*
- void **toggle** ()  
*Toggle activation.*
- void **set\_auto\_bind** (bool value=true)  
*Set auto bind.*
- bool **auto\_bind** () const  
*Check if auto bind is enabled.*
- bool **ready** () const  
*Check if pipeline is ready.*
- VkPipeline **get** () const  
*Get the pipeline.*
- device::ptr **get\_device** ()  
*Get the device.*
- pipeline\_layout::s\_ptr **get\_layout** () const  
*Get the layout.*
- void **set\_layout** (pipeline\_layout::s\_ptr const &value)  
*Set the layout.*

**Public Member Functions inherited from [lava::entity](#)**

- **entity** ()  
*Construct a new entity.*
- **id::ref get\_id** () const  
*Get the id of entity.*

**Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)**

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

**Public Member Functions inherited from [lava::interface](#)**

- virtual **~interface** ()=default  
*Destroy the interface.*

**Static Public Member Functions**

- static **s\_ptr make** ([device::ptr device](#), VkPipelineCache pipeline\_cache=0)  
*Make a new compute pipeline.*

**Additional Inherited Members****Public Attributes inherited from [lava::pipeline](#)**

- **process\_func on\_process**  
*Called on pipeline process.*

**Protected Member Functions inherited from [lava::pipeline](#)****Protected Attributes inherited from [lava::pipeline](#)**

- **device::ptr m\_device** = nullptr  
*Vulkan device.*
- VkPipeline **m\_vk\_pipeline** = VK\_NULL\_HANDLE  
*Vulkan pipeline.*
- VkPipelineCache **m\_pipeline\_cache** = VK\_NULL\_HANDLE  
*Vulkan pipeline cache.*
- **pipeline\_layout::s\_ptr m\_layout**  
*Pipeline layout.*

### 4.15.1 Detailed Description

Compute pipeline.

### 4.15.2 Member Function Documentation

#### 4.15.2.1 bind()

```
void lava::compute_pipeline::bind (
    VkCommandBuffer cmdBuffer) [override], [virtual]
```

Bind the pipeline.

##### Parameters

<i>cmdBuffer</i>	Command buffer
------------------	----------------

Implements [lava::pipeline](#).

#### 4.15.2.2 copy\_from()

```
void lava::compute_pipeline::copy_from (
    s_ptr const & source) [inline]
```

Copy configuration from source.

##### Parameters

<i>source</i>	Compute pipeline
---------------	------------------

#### 4.15.2.3 copy\_to()

```
void lava::compute_pipeline::copy_to (
    compute_pipeline * target) const
```

Copy configuration to target pipeline.

##### Parameters

<i>target</i>	Compute pipeline
---------------	------------------

#### 4.15.2.4 get\_shader\_stage()

```
shader_stage::s_ptr const & lava::compute_pipeline::get_shader_stage () const [inline]
```

Get the shader stage.

##### Returns

shader\_stage::s\_ptr const& Shader state



#### 4.15.2.5 make()

```
static s_ptr lava::compute_pipeline::make (  
    device::ptr device,  
    VkPipelineCache pipeline_cache = 0) [inline], [static]
```

Make a new compute pipeline.

## Parameters

<i>device</i>	Vulkan device
<i>pipeline_cache</i>	Pipeline cache

## Returns

s\_ptr Shared pointer to compute pipeline

**4.15.2.6 set()**

```
void lava::compute_pipeline::set (
    shader_stage::s_ptr const & stage) [inline]
```

Set shader stage.

## Parameters

<i>stage</i>	Shader state
--------------	--------------

**4.15.2.7 set\_shader\_stage()**

```
bool lava::compute_pipeline::set_shader_stage (
    c_data::ref data,
    VkShaderStageFlagBits stage)
```

Set shader stage.

## Parameters

<i>data</i>	Shader data
<i>stage</i>	Shader stage flag bits

## Returns

Set was successful or failed

The documentation for this struct was generated from the following file:

- liblava/block/[compute\\_pipeline.hpp](#)

**4.16 lava::imgui::config Struct Reference**

ImGui configuration.

```
#include <imgui.hpp>
```

### Public Attributes

- [data](#) **font\_data**  
*Font data.*
- [r32](#) **font\_size** = [default\\_imgui\\_font\\_size](#)  
*Font size.*
- `std::shared_ptr< ImGuiStyle >` **style**  
*Font style.*
- [icon\\_font](#) **icon**  
*Font icon settings.*
- `std::filesystem::path` **ini\_file\_dir**  
*ImGui state file path.*
- [i32](#) **flags** = 0  
*ImGuiConfigFlags.*

#### 4.16.1 Detailed Description

ImGui configuration.

The documentation for this struct was generated from the following file:

- [liblava/app/imgui.hpp](#)

## 4.17 lava::log::config Struct Reference

Log configuration.

```
#include <log.hpp>
```

### Public Attributes

- [name](#) **logger** = [\\_lava\\_](#)  
*Logger name.*
- [name](#) **file** = "lava.log"  
*Log file.*
- [i32](#) **level** = [undef](#)  
*Log level.*
- `bool` **debug** = false  
*Log to console, else file.*

#### 4.17.1 Detailed Description

Log configuration.

The documentation for this struct was generated from the following file:

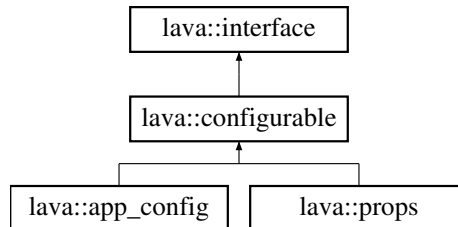
- [liblava/util/log.hpp](#)

## 4.18 lava::configurable Struct Reference

Configurable interface.

```
#include <json.hpp>
```

Inheritance diagram for lava::configurable:



### Public Member Functions

- virtual void [set\\_json](#) (json\_ref j)=0  
*Set json config.*
- virtual json [get\\_json](#) () const =0  
*Get json config.*

### Public Member Functions inherited from [lava::interface](#)

- virtual  $\sim$ **interface** ()=default  
*Destroy the interface.*

#### 4.18.1 Detailed Description

Configurable interface.

#### 4.18.2 Member Function Documentation

##### 4.18.2.1 [get\\_json\(\)](#)

```
virtual json lava::configurable::get_json () const [pure virtual]
```

Get json config.

##### Returns

json Json file

Implemented in [lava::app\\_config](#), and [lava::props](#).

##### 4.18.2.2 [set\\_json\(\)](#)

```
virtual void lava::configurable::set_json (
    json_ref j) [pure virtual]
```

Set json config.

## Parameters

<i>j</i>	Json file
----------	-----------

Implemented in [lava::app\\_config](#), and [lava::props](#).

The documentation for this struct was generated from the following file:

- liblava/file/json.hpp

## 4.19 lava::render\_pipeline::create\_info Struct Reference

Render pipeline create information.

```
#include <render_pipeline.hpp>
```

### Public Attributes

- VkPipelineVertexInputStateCreateInfo **vertex\_input\_state**  
*Vertex input state.*
- VkPipelineInputAssemblyStateCreateInfo **input\_assembly\_state**  
*Input assembly state.*
- VkPipelineViewportStateCreateInfo **viewport\_state**  
*Viewport state.*
- VkPipelineMultisampleStateCreateInfo **multisample\_state**  
*Multisample state.*
- VkPipelineDepthStencilStateCreateInfo **depth\_stencil\_state**  
*Depth stencil state.*
- VkPipelineRasterizationStateCreateInfo **rasterization\_state**  
*Rasterization state.*

### 4.19.1 Detailed Description

Render pipeline create information.

The documentation for this struct was generated from the following file:

- liblava/block/[render\\_pipeline.hpp](#)

## 4.20 lava::device::create\_param Struct Reference

Device create parameters.

```
#include <device.hpp>
```

## Public Types

- using **ref** = [create\\_param](#) const&  
*Reference to device create parameters.*

## Public Member Functions

- void **add\_swapchain\_extension** ()  
*Add swapchain extension.*
- void **add\_portability\_subset\_extension** ()  
*Add portability subset extension.*
- void **set\_default\_queues** ()  
*Set the default queues.*
- void **set\_all\_queues** ()  
*Set the all queues.*
- bool **add\_queue** (VkQueueFlags flags, [r32](#) priority=1.f)  
*Add queue.*
- bool **add\_queues** (VkQueueFlags flags, [ui32](#) count, [r32](#) priority=1.f)  
*Add queues.*
- bool **add\_dedicated\_queues** ([r32](#) priority=1.f)  
*Add all dedicated queues.*
- [verify\\_queues\\_result](#) **verify\_queues** () const  
*Verify queues.*

## Public Attributes

- [physical\\_device\\_c\\_ptr](#) **physical\_device** = nullptr  
*Physical device.*
- VmaAllocatorCreateFlags **vma\_flags** = 0  
*VMA flags.*
- [names](#) **extensions**  
*List of extensions to enable.*
- VkPhysicalDeviceFeatures **features** {}  
*List of physical device features to enable.*
- bool **has\_features\_2** = false  
*Must be true if .next points to a VkPhysicalDevice2 instance.*
- void const \* **next** = nullptr  
*Create parameter next pointer (pNext)*
- [queue\\_family\\_info::list](#) **queue\_family\_infos**  
*List of queue famiy infos.*

### 4.20.1 Detailed Description

Device create parameters.

### 4.20.2 Member Function Documentation

#### 4.20.2.1 add\_dedicated\_queues()

```
bool lava::device::create_param::add_dedicated_queues (
    r32 priority = 1.f)
```

Add all dedicated queues.

## Parameters

<i>priority</i>	Priority for queues
-----------------	---------------------

## Returns

Add was successful or failed

#### 4.20.2.2 add\_queue()

```
bool lava::device::create_param::add_queue (
    VkQueueFlags flags,
    r32 priority = 1.f) [inline]
```

Add queue.

## Parameters

<i>flags</i>	Queue flags
<i>priority</i>	Priority for queue

## Returns

Add was successful or failed

#### 4.20.2.3 add\_queues()

```
bool lava::device::create_param::add_queues (
    VkQueueFlags flags,
    ui32 count,
    r32 priority = 1.f)
```

Add queues.

## Parameters

<i>flags</i>	Queue flags
<i>count</i>	Number of queues
<i>priority</i>	Priority for queues

## Returns

Add was successful or failed

#### 4.20.2.4 verify\_queues()

```
verify_queues_result lava::device::create_param::verify_queues () const
```

Verify queues.

##### Returns

verify\_queues\_result Verification result

The documentation for this struct was generated from the following file:

- liblava/base/[device.hpp](#)

## 4.21 lava::instance::create\_param Struct Reference

Instance create parameters.

```
#include <instance.hpp>
```

### Public Types

- using **ref** = [create\\_param](#) const&  
*Reference to instance create parameters.*

### Public Attributes

- [names](#) **layers** {}  
*List of layers to enable.*
- [names](#) **extensions** {}  
*List of extensions to enable.*

#### 4.21.1 Detailed Description

Instance create parameters.

The documentation for this struct was generated from the following file:

- liblava/base/[instance.hpp](#)

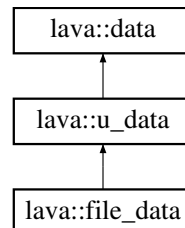


## 4.22 lava::data Struct Reference

Data wrapper.

```
#include <data.hpp>
```

Inheritance diagram for lava::data:



### Public Types

- enum class [mode](#) : index { **alloc** = 0 , **no\_alloc** }  
*Data modes.*
- using **ref** = [data](#) const&  
*Reference to data wrapper.*
- using **ptr** = char\*  
*Data pointer.*
- using **c\_ptr** = char const\*  
*Const data pointer.*

### Public Member Functions

- **data** ()=default  
*Construct a new data.*
- [data](#) (auto \*[addr](#), [size\\_t](#) [size](#))  
*Construct a new data.*
- bool [set](#) ([size\\_t](#) [length](#), [mode](#) [mode](#)=[mode::alloc](#))  
*Set and allocate data by length.*
- bool [allocate](#) ()  
*Allocate data.*
- void **deallocate** ()  
*Deallocate data.*
- [ptr](#) [end](#) () const  
*Pointer to end of data.*

### Static Public Member Functions

- static [ptr](#) [as\\_ptr](#) (auto \*[value](#))  
*Cast to data pointer.*
- static [c\\_ptr](#) [as\\_c\\_ptr](#) (auto \*[value](#))  
*Cast to const data pointer.*

## Public Attributes

- `ptr addr = nullptr`  
*Pointer address.*
- `size_t size = 0`  
*Size of data.*
- `size_t alignment = 0`  
*Data alignment.*

### 4.22.1 Detailed Description

Data wrapper.

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 data()

```
lava::data::data (
    auto * addr,
    size_t size) [inline]
```

Construct a new data.

#### Parameters

<i>addr</i>	Data pointer
<i>size</i>	Size of data

### 4.22.3 Member Function Documentation

#### 4.22.3.1 allocate()

```
bool lava::data::allocate () [inline]
```

Allocate data.

#### Returns

Allocate was successful or failed

#### 4.22.3.2 as\_c\_ptr()

```
static c_ptr lava::data::as_c_ptr (
    auto * value) [inline], [static]
```

Cast to const data pointer.

## Parameters

<i>value</i>	Value to cast
--------------	---------------

## Returns

c\_ptr Const data pointer

### 4.22.3.3 as\_ptr()

```
static ptr lava::data::as_ptr (  
    auto * value) [inline], [static]
```

Cast to data pointer.

## Parameters

<i>value</i>	Value to cast
--------------	---------------

## Returns

ptr Data pointer

### 4.22.3.4 end()

```
ptr lava::data::end () const [inline]
```

Pointer to end of data.

## Returns

ptr Pointer to end

### 4.22.3.5 set()

```
bool lava::data::set (  
    size_t length,  
    mode mode = mode::alloc) [inline]
```

Set and allocate data by length.

## Parameters

<i>length</i>	Length of data
<i>mode</i>	Data mode

## Returns

Allocate was successful or failed (mode: alloc)

The documentation for this struct was generated from the following file:

- [liblava/core/data.hpp](#)

## 4.23 `lava::data_provider` Struct Reference

Data provider.

```
#include <data.hpp>
```

### Public Types

- using **`alloc_func`** = `std::function<data::ptr(size_t, size_t)>`  
*Allocation function.*
- using **`free_func`** = `std::function<void()>`  
*Free function.*
- using **`realloc_func`** = `std::function<data::ptr(data::ptr, size_t, size_t)>`  
*Reallocation function.*

### Public Attributes

- [alloc\\_func](#) **`on_alloc`**  
*Called on allocation.*
- [free\\_func](#) **`on_free`**  
*Called on free.*
- [realloc\\_func](#) **`on_realloc`**  
*Called on reallocation.*

### 4.23.1 Detailed Description

Data provider.

The documentation for this struct was generated from the following file:

- `liblava/core/data.hpp`

## 4.24 `lava::instance::debug_config` Struct Reference

Debug configuration.

```
#include <instance.hpp>
```

### Public Types

- using **`ref`** = [debug\\_config](#) const&  
*Reference to debug configuration.*

### Public Attributes

- bool **validation** = false  
*Validation.*
- bool **render\_doc** = false  
*Renderdoc.*
- bool **verbose** = false  
*Verbose logging.*
- bool **utils** = false  
*Debug utils.*

#### 4.24.1 Detailed Description

Debug configuration.

The documentation for this struct was generated from the following file:

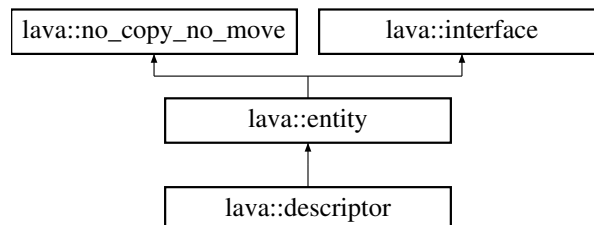
- liblava/base/[instance.hpp](#)

## 4.25 lava::descriptor Struct Reference

Descriptor.

```
#include <descriptor.hpp>
```

Inheritance diagram for lava::descriptor:



### Classes

- struct [binding](#)  
*Descriptor binding.*
- struct [pool](#)  
*Descriptor pool.*

### Public Types

- using **s\_ptr** = std::shared\_ptr<[descriptor](#)>  
*Shared pointer to descriptor.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of descriptors.*

## Public Member Functions

- void [add\\_binding](#) ([index binding](#), VkDescriptorType descriptor\_type, VkShaderStageFlags stage\_flags)  
*Add binding.*
- void **clear\_bindings** ()  
*Clear bindings.*
- void [add](#) ([binding::s\\_ptr](#) const &[binding](#))  
*Add binding.*
- bool [create](#) ([device::ptr](#) device)  
*Create a new descriptor.*
- void **destroy** ()  
*Destroy the descriptor.*
- [ui32 get\\_binding\\_count](#) () const  
*Get the binding count.*
- [binding::s\\_list](#) const & [get\\_bindings](#) ()  
*Get the bindings.*
- VkDescriptorSetLayout [get](#) () const  
*Get descriptor set layout.*
- [device::ptr](#) [get\\_device](#) ()  
*Get the device.*
- VkDescriptorSet [allocate\\_set](#) (VkDescriptorPool [pool](#))  
*Allocate descriptor set.*
- VkDescriptorSet [allocate](#) (VkDescriptorPool [pool](#))
- bool [deallocate\\_set](#) (VkDescriptorSet &descriptor\_set, VkDescriptorPool [pool](#))  
*Deallocate descriptor set.*
- bool [deallocate](#) (VkDescriptorSet &descriptor\_set, VkDescriptorPool [pool](#))
- [VkDescriptorSets](#) [allocate\\_sets](#) ([ui32](#) size, VkDescriptorPool [pool](#))  
*Allocate descriptor sets.*
- [VkDescriptorSets](#) [allocate](#) ([ui32](#) size, VkDescriptorPool [pool](#))
- bool [deallocate\\_sets](#) ([VkDescriptorSets](#) &descriptor\_sets, VkDescriptorPool [pool](#))  
*Deallocate descriptor sets.*
- bool [deallocate](#) ([VkDescriptorSets](#) &descriptor\_sets, VkDescriptorPool [pool](#))

## Public Member Functions inherited from [lava::entity](#)

- **entity** ()  
*Construct a new entity.*
- [id::ref](#) [get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default  
*Destroy the interface.*

### Static Public Member Functions

- static `s_ptr make()`  
*Make a new descriptor.*

## 4.25.1 Detailed Description

Descriptor.

## 4.25.2 Member Function Documentation

### 4.25.2.1 add()

```
void lava::descriptor::add (  
    binding::s\_ptr const & binding) [inline]
```

Add binding.

#### Parameters

<i>binding</i>	Descriptor binding
----------------	--------------------

### 4.25.2.2 add\_binding()

```
void lava::descriptor::add_binding (  
    index binding,  
    VkDescriptorType descriptor_type,  
    VkShaderStageFlags stage_flags)
```

Add binding.

#### Parameters

<i>binding</i>	Index of binding
<i>descriptor_type</i>	Descriptor type
<i>stage_flags</i>	Shader stage flags

**4.25.2.3 allocate()** [1/2]

```
VkDescriptorSets lava::descriptor::allocate (
    ui32 size,
    VkDescriptorPool pool) [inline]
```

See also

[allocate\\_sets](#)

**4.25.2.4 allocate()** [2/2]

```
VkDescriptorSet lava::descriptor::allocate (
    VkDescriptorPool pool) [inline]
```

See also

[allocate\\_set](#)

**4.25.2.5 allocate\_set()**

```
VkDescriptorSet lava::descriptor::allocate_set (
    VkDescriptorPool pool)
```

Allocate descriptor set.

Parameters

<i>pool</i>	Descriptor pool
-------------	-----------------

Returns

VkDescriptorSet Descriptor set

**4.25.2.6 allocate\_sets()**

```
VkDescriptorSets lava::descriptor::allocate_sets (
    ui32 size,
    VkDescriptorPool pool)
```

Allocate descriptor sets.

Parameters

<i>size</i>	Number of sets
<i>pool</i>	Descriptor pool

Returns

VkDescriptorSets List of descriptor sets

**4.25.2.7 create()**

```
bool lava::descriptor::create (
    device::ptr device)
```

Create a new descriptor.



## Parameters

<i>device</i>	Vulkan device
---------------	---------------

## Returns

Create was successful or failed

**4.25.2.8 deallocate()** [1/2]

```
bool lava::descriptor::deallocate (
    VkDescriptorSet & descriptor_set,
    VkDescriptorPool pool) [inline]
```

## See also

[deallocate\\_set](#)

**4.25.2.9 deallocate()** [2/2]

```
bool lava::descriptor::deallocate (
    VkDescriptorSets & descriptor_sets,
    VkDescriptorPool pool) [inline]
```

## See also

[deallocate\\_sets](#)

**4.25.2.10 deallocate\_set()**

```
bool lava::descriptor::deallocate_set (
    VkDescriptorSet & descriptor_set,
    VkDescriptorPool pool)
```

Deallocate descriptor set.

## Parameters

<i>descriptor_set</i>	Descriptor set
<i>pool</i>	Descriptor pool

## Returns

Deallocate was successful or failed

**4.25.2.11 deallocate\_sets()**

```
bool lava::descriptor::deallocate_sets (
    VkDescriptorSets & descriptor_sets,
    VkDescriptorPool pool)
```

Deallocate descriptor sets.

**Parameters**

<i>descriptor_sets</i>	List of descriptor sets
<i>pool</i>	Descriptor pool

**Returns**

Deallocate was successful or failed

**4.25.2.12 get()**

```
VkDescriptorSetLayout lava::descriptor::get () const [inline]
```

Get descriptor set layout.

**Returns**

VkDescriptorSetLayout Vulkan descriptor set layout

**4.25.2.13 get\_binding\_count()**

```
ui32 lava::descriptor::get_binding_count () const [inline]
```

Get the binding count.

**Returns**

ui32 Number of bindings

**4.25.2.14 get\_bindings()**

```
binding::s_list const & lava::descriptor::get_bindings () [inline]
```

Get the bindings.

**Returns**

[binding::s\\_list](#) const& List of bindings

**4.25.2.15 get\_device()**

```
device::ptr lava::descriptor::get_device () [inline]
```

Get the device.

**Returns**

[device::ptr](#) Vulkan device

## 4.25.2.16 make()

```
static s_ptr lava::descriptor::make () [inline], [static]
```

Make a new descriptor.

## Returns

s\_ptr Shared pointer to descriptor

The documentation for this struct was generated from the following file:

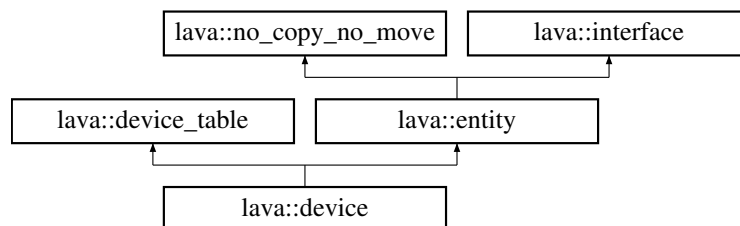
- liblava/block/[descriptor.hpp](#)

## 4.26 lava::device Struct Reference

Vulkan device.

```
#include <device.hpp>
```

Inheritance diagram for lava::device:



## Classes

- struct [create\\_param](#)  
Device create parameters.

## Public Types

- using **ptr** = [device\\*](#)  
Pointer to device.
- using **c\_ptr** = [device](#) const\*  
Const pointer to device.
- using **s\_ptr** = std::shared\_ptr<[device](#)>  
Shared pointer to a device.
- using **s\_list** = std::vector<[s\\_ptr](#)>  
List of devices.
- using **physical\_device\_c\_ptr** = [physical\\_device](#) const\*  
Const pointer to a physical device.

## Public Member Functions

- `~device ()`  
*Destroy the device.*
- `bool create (create_param::ref param)`  
*Create a new device.*
- `void destroy ()`  
*Destroy the device.*
- `queue::ref get_graphics_queue (index index=0) const`  
*Get a graphics queue by index.*
- `queue::ref graphics_queue (index index=0) const`
- `queue::ref get_compute_queue (index index=0) const`  
*Get a compute queue by index.*
- `queue::ref compute_queue (index index=0) const`
- `queue::ref get_transfer_queue (index index=0) const`  
*Get a transfer queue by index.*
- `queue::ref transfer_queue (index index=0) const`
- `queue::list const & get_graphics_queues () const`  
*Get the list of graphics queues.*
- `queue::list const & graphics_queues () const`
- `queue::list const & get_compute_queues () const`  
*Get the list of compute queues.*
- `queue::list const & compute_queues () const`
- `queue::list const & get_transfer_queues () const`  
*Get the list of transfer queues.*
- `queue::list const & transfer_queues () const`
- `queue::list const & get_queues () const`  
*Get all queues.*
- `queue::list const & queues () const`
- `VkDevice get () const`  
*Get the Vulkan device.*
- `VolkDeviceTable const & call () const`  
*Get the Volk device table.*
- `bool wait_for_idle () const`  
*Wait for idle.*
- `physical_device_c_ptr get_physical_device () const`  
*Get the physical device.*
- `VkPhysicalDevice get_vk_physical_device () const`  
*Get the Vulkan physical device.*
- `VkPhysicalDeviceFeatures const & get_features () const`  
*Get the physical device features.*
- `VkPhysicalDeviceProperties const & get_properties () const`  
*Get the physical device properties.*
- `bool surface_supported (VkSurfaceKHR surface) const`  
*Check if surface is supported by this device.*
- `void set_allocator (allocator::s_ptr value)`  
*Set the allocator for this device.*
- `allocator::s_ptr get_allocator ()`  
*Get the allocator of this device.*
- `VmaAllocator alloc () const`  
*Get the VMA allocator.*

Public Member Functions inherited from [lava::device\\_table](#)

- void **load\_table** ()  
*Load device table.*
- [vk\\_result vkCreateImageView](#) (const VkImageViewCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkImageView \*pView)
- [vk\\_result vkCreateImageView](#) (const VkImageViewCreateInfo \*pCreateInfo, VkImageView \*pView)
- [vk\\_result vkCreateSampler](#) (const VkSamplerCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSampler \*pSampler)
- [vk\\_result vkCreateSampler](#) (const VkSamplerCreateInfo \*pCreateInfo, VkSampler \*pSampler)
- [vk\\_result vkCreateShaderModule](#) (const VkShaderModuleCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkShaderModule \*pShaderModule)
- [vk\\_result vkCreateShaderModule](#) (const VkShaderModuleCreateInfo \*pCreateInfo, VkShaderModule \*pShaderModule)
- [vk\\_result vkCreateFence](#) (const VkFenceCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkFence \*pFence)
- [vk\\_result vkCreateFence](#) (const VkFenceCreateInfo \*pCreateInfo, VkFence \*pFence)
- [vk\\_result vkCreateSemaphore](#) (const VkSemaphoreCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSemaphore \*pSemaphore)
- [vk\\_result vkCreateSemaphore](#) (const VkSemaphoreCreateInfo \*pCreateInfo, VkSemaphore \*pSemaphore)
- [vk\\_result vkWaitForFences](#) (uint32\_t fenceCount, const VkFence \*pFences, VkBool32 waitAll, uint64\_t timeout)
- [vk\\_result vkResetFences](#) (uint32\_t fenceCount, const VkFence \*pFences)
- [vk\\_result vkQueueSubmit](#) (VkQueue [queue](#), uint32\_t submitCount, const VkSubmitInfo \*pSubmits, VkFence fence)
- [vk\\_result vkAcquireNextImageKHR](#) (VkSwapchainKHR [swapchain](#), uint64\_t timeout, VkSemaphore semaphore, VkFence fence, uint32\_t \*pImageIndex)
- [vk\\_result vkQueuePresentKHR](#) (VkQueue [queue](#), const VkPresentInfoKHR \*pPresentInfo)
- [vk\\_result vkCreateSwapchainKHR](#) (const VkSwapchainCreateInfoKHR \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSwapchainKHR \*pSwapchain)
- [vk\\_result vkCreateSwapchainKHR](#) (const VkSwapchainCreateInfoKHR \*pCreateInfo, VkSwapchainKHR \*pSwapchain)
- void [vkDestroySwapchainKHR](#) (VkSwapchainKHR [swapchain](#), const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- [vk\\_result vkGetSwapchainImagesKHR](#) (VkSwapchainKHR [swapchain](#), uint32\_t \*pSwapchainImageCount, VkImage \*pSwapchainImages)
- [vk\\_result vkCreateCommandPool](#) (const VkCommandPoolCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkCommandPool \*pCommandPool)
- [vk\\_result vkCreateCommandPool](#) (const VkCommandPoolCreateInfo \*pCreateInfo, VkCommandPool \*pCommandPool)
- [vk\\_result vkCreateCommandPool](#) ([index](#) queue\_family, VkCommandPool \*pCommandPool)
- [vk\\_result vkAllocateCommandBuffers](#) (const VkCommandBufferAllocateInfo \*pAllocateInfo, VkCommandBuffer \*pCommandBuffers)
- [vk\\_result vkAllocateCommandBuffers](#) (VkCommandPool commandPool, uint32\_t commandBufferCount, VkCommandBuffer \*pCommandBuffers, VkCommandBufferLevel level=VK\_COMMAND\_BUFFER\_LEVEL\_PRIMARY)
- void [vkDestroyImageView](#) (VkImageView imageView, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- void [vkDestroyFence](#) (VkFence fence, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- void [vkDestroySemaphore](#) (VkSemaphore semaphore, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- void [vkFreeCommandBuffers](#) (VkCommandPool commandPool, uint32\_t commandBufferCount, const VkCommandBuffer \*pCommandBuffers)
- void [vkDestroyCommandPool](#) (VkCommandPool commandPool, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- void [vkDestroySampler](#) (VkSampler sampler, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\)](#).alloc())
- void [vkUpdateDescriptorSets](#) (uint32\_t descriptorWriteCount, const VkWriteDescriptorSet \*pDescriptorWrites, uint32\_t descriptorCopyCount=0, const VkCopyDescriptorSet \*pDescriptorCopies=nullptr)

- `template<std::size_t SIZE>`  
void `vkUpdateDescriptorSets` (`std::array< VkWriteDescriptorSet, SIZE > const &descriptor_writes`)
- `template<std::size_t SIZE>`  
void `vkUpdateDescriptorSets` (`std::array< VkCopyDescriptorSet, SIZE > const &descriptor_copies`)
- `template<std::size_t WRITE_SIZE, std::size_t COPY_SIZE>`  
void `vkUpdateDescriptorSets` (`std::array< VkWriteDescriptorSet, WRITE_SIZE > const &descriptor_writes`,  
`std::array< VkCopyDescriptorSet, COPY_SIZE > const &descriptor_copies`)
- void `vkUpdateDescriptorSets` (`std::initializer_list< VkWriteDescriptorSet > descriptor_writes`)
- void `vkUpdateDescriptorSets` (`std::initializer_list< VkCopyDescriptorSet > descriptor_copies`)
- void `vkUpdateDescriptorSets` (`std::initializer_list< VkWriteDescriptorSet > descriptor_writes`, `std::initializer_list< VkCopyDescriptorSet > descriptor_copies`)

## Public Member Functions inherited from `lava::entity`

- `entity ()`  
*Construct a new entity.*
- `id::ref get_id () const`  
*Get the id of entity.*

## Public Member Functions inherited from `lava::no_copy_no_move`

- `no_copy_no_move ()=default`  
*Construct a new object.*
- `no_copy_no_move (no_copy_no_move const &)=delete`  
*No copy.*
- void `operator= (no_copy_no_move const &)=delete`  
*No move.*

## Public Member Functions inherited from `lava::interface`

- virtual `~interface ()=default`  
*Destroy the interface.*

## Static Public Member Functions

- static `s_ptr make ()`  
*Make a new device.*

## Additional Inherited Members

## Public Attributes inherited from `lava::device_table`

- `VkDevice vk_device = nullptr`  
*Vulkan device.*
- `VolkDeviceTable table = {}`  
*Volk device table.*

## 4.26.1 Detailed Description

Vulkan device.

## 4.26.2 Member Function Documentation

### 4.26.2.1 alloc()

```
VmaAllocator lava::device::alloc () const [inline]
```

Get the VMA allocator.

Returns

VmaAllocator VMA allocator

### 4.26.2.2 call()

```
VolkDeviceTable const & lava::device::call () const [inline]
```

Get the Volk device table.

Returns

VolkDeviceTable const& Volk device table

### 4.26.2.3 compute\_queue()

```
queue::ref lava::device::compute_queue (  
    index index = 0) const [inline]
```

See also

[get\\_compute\\_queue](#)

### 4.26.2.4 compute\_queues()

```
queue::list const & lava::device::compute_queues () const [inline]
```

See also

[get\\_compute\\_queues](#)

### 4.26.2.5 create()

```
bool lava::device::create (  
    create_param::ref param)
```

Create a new device.

## Parameters

<i>param</i>	Create parameters
--------------	-------------------

## Returns

Create was successful or failed

**4.26.2.6 get()**

```
VkDevice lava::device::get () const [inline]
```

Get the Vulkan device.

## Returns

VkDevice Vulkan device

**4.26.2.7 get\_allocator()**

```
allocator::s_ptr lava::device::get_allocator () [inline]
```

Get the allocator of this device.

## Returns

[allocator::s\\_ptr](#) Allocator

**4.26.2.8 get\_compute\_queue()**

```
queue::ref lava::device::get_compute_queue (  
    index index = 0) const [inline]
```

Get a compute queue by index.

## Parameters

<i>index</i>	Index of queue
--------------	----------------

## Returns

[queue::ref](#) Compute queue



#### 4.26.2.9 get\_compute\_queues()

```
queue::list const & lava::device::get_compute_queues () const [inline]
```

Get the list of compute queues.

Returns

queue::list const& Compute queues

#### 4.26.2.10 get\_features()

```
VkPhysicalDeviceFeatures const & lava::device::get_features () const
```

Get the physical device features.

Returns

VkPhysicalDeviceFeatures const& Features

#### 4.26.2.11 get\_graphics\_queue()

```
queue::ref lava::device::get_graphics_queue (  
    index index = 0) const [inline]
```

Get a graphics queue by index.

Parameters

<i>index</i>	Index of queue
--------------	----------------

Returns

queue::ref Graphics queue

#### 4.26.2.12 get\_graphics\_queues()

```
queue::list const & lava::device::get_graphics_queues () const [inline]
```

Get the list of graphics queues.

Returns

queue::list const& Graphics queues

**4.26.2.13 get\_physical\_device()**

```
physical_device_c_ptr lava::device::get_physical_device () const [inline]
```

Get the physical device.

**Returns**

physical\_device\_c\_ptr Physical device

**4.26.2.14 get\_properties()**

```
VkPhysicalDeviceProperties const & lava::device::get_properties () const
```

Get the physical device properties.

**Returns**

VkPhysicalDeviceProperties const& Properties

**4.26.2.15 get\_queues()**

```
queue::list const & lava::device::get_queues () const [inline]
```

Get all queues.

**Returns**

queue::list const& List of all queues

**4.26.2.16 get\_transfer\_queue()**

```
queue::ref lava::device::get_transfer_queue (
    index index = 0) const [inline]
```

Get a transfer queue by index.

**Parameters**

<i>index</i>	Index of queue
--------------	----------------

**Returns**

queue::ref Transfer queue

#### 4.26.2.17 get\_transfer\_queues()

```
queue::list const & lava::device::get_transfer_queues () const [inline]
```

Get the list of transfer queues.

Returns

[queue::list](#) const& Transfer queues

#### 4.26.2.18 get\_vk\_physical\_device()

```
VkPhysicalDevice lava::device::get_vk_physical_device () const
```

Get the Vulkan physical device.

Returns

VkPhysicalDevice Vulkan physical device

#### 4.26.2.19 graphics\_queue()

```
queue::ref lava::device::graphics_queue (  
    index index = 0) const [inline]
```

See also

[get\\_graphics\\_queue](#)

#### 4.26.2.20 graphics\_queues()

```
queue::list const & lava::device::graphics_queues () const [inline]
```

See also

[get\\_graphics\\_queues](#)

#### 4.26.2.21 make()

```
static s_ptr lava::device::make () [inline], [static]
```

Make a new device.

Returns

s\_ptr Shared pointer to device

#### 4.26.2.22 queues()

```
queue::list const & lava::device::queues () const [inline]
```

See also

[get\\_queues](#)

#### 4.26.2.23 set\_allocator()

```
void lava::device::set_allocator (  
    allocator::s_ptr value) [inline]
```

Set the allocator for this device.

## Parameters

<i>value</i>	Allocator
--------------	-----------

**4.26.2.24 surface\_supported()**

```
bool lava::device::surface_supported (
    VkSurfaceKHR surface) const
```

Check if surface is supported by this device.

## Parameters

<i>surface</i>	Surface to check
----------------	------------------

## Returns

Surface is supported or not

**4.26.2.25 transfer\_queue()**

```
queue::ref lava::device::transfer_queue (
    index index = 0) const [inline]
```

## See also

[get\\_transfer\\_queue](#)

**4.26.2.26 transfer\_queues()**

```
queue::list const & lava::device::transfer_queues () const [inline]
```

## See also

[get\\_transfer\\_queues](#)

**4.26.2.27 wait\_for\_idle()**

```
bool lava::device::wait_for_idle () const [inline]
```

Wait for idle.

## Returns

Wait was successful or failed

The documentation for this struct was generated from the following file:

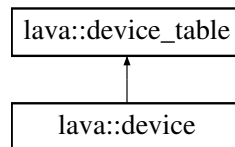
- liblava/base/[device.hpp](#)

## 4.27 lava::device\_table Struct Reference

Device functions.

```
#include <device_table.hpp>
```

Inheritance diagram for lava::device\_table:



### Public Member Functions

- void **load\_table** ()  
Load device table.
- [vk\\_result vkCreateImageView](#) (const VkImageViewCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkImageView \*pView)
- [vk\\_result vkCreateImageView](#) (const VkImageViewCreateInfo \*pCreateInfo, VkImageView \*pView)
- [vk\\_result vkCreateSampler](#) (const VkSamplerCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSampler \*pSampler)
- [vk\\_result vkCreateSampler](#) (const VkSamplerCreateInfo \*pCreateInfo, VkSampler \*pSampler)
- [vk\\_result vkCreateShaderModule](#) (const VkShaderModuleCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkShaderModule \*pShaderModule)
- [vk\\_result vkCreateShaderModule](#) (const VkShaderModuleCreateInfo \*pCreateInfo, VkShaderModule \*pShaderModule)
- [vk\\_result vkCreateFence](#) (const VkFenceCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkFence \*pFence)
- [vk\\_result vkCreateFence](#) (const VkFenceCreateInfo \*pCreateInfo, VkFence \*pFence)
- [vk\\_result vkCreateSemaphore](#) (const VkSemaphoreCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSemaphore \*pSemaphore)
- [vk\\_result vkCreateSemaphore](#) (const VkSemaphoreCreateInfo \*pCreateInfo, VkSemaphore \*pSemaphore)
- [vk\\_result vkWaitForFences](#) (uint32\_t fenceCount, const VkFence \*pFences, VkBool32 waitAll, uint64\_t timeout)
- [vk\\_result vkResetFences](#) (uint32\_t fenceCount, const VkFence \*pFences)
- [vk\\_result vkQueueSubmit](#) (VkQueue [queue](#), uint32\_t submitCount, const VkSubmitInfo \*pSubmits, VkFence fence)
- [vk\\_result vkAcquireNextImageKHR](#) (VkSwapchainKHR [swapchain](#), uint64\_t timeout, VkSemaphore semaphore, VkFence fence, uint32\_t \*pImageIndex)
- [vk\\_result vkQueuePresentKHR](#) (VkQueue [queue](#), const VkPresentInfoKHR \*pPresentInfo)
- [vk\\_result vkCreateSwapchainKHR](#) (const VkSwapchainCreateInfoKHR \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkSwapchainKHR \*pSwapchain)
- [vk\\_result vkCreateSwapchainKHR](#) (const VkSwapchainCreateInfoKHR \*pCreateInfo, VkSwapchainKHR \*pSwapchain)
- void [vkDestroySwapchainKHR](#) (VkSwapchainKHR [swapchain](#), const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- [vk\\_result vkGetSwapchainImagesKHR](#) (VkSwapchainKHR [swapchain](#), uint32\_t \*pSwapchainImageCount, VkImage \*pSwapchainImages)
- [vk\\_result vkCreateCommandPool](#) (const VkCommandPoolCreateInfo \*pCreateInfo, const VkAllocationCallbacks \*pAllocator, VkCommandPool \*pCommandPool)
- [vk\\_result vkCreateCommandPool](#) (const VkCommandPoolCreateInfo \*pCreateInfo, VkCommandPool \*pCommandPool)

- [vk\\_result vkCreateCommandPool](#) ([index](#) queue\_family, VkCommandPool \*pCommandPool)
- [vk\\_result vkAllocateCommandBuffers](#) (const VkCommandBufferAllocateInfo \*pAllocateInfo, VkCommandBuffer \*pCommandBuffers)
- [vk\\_result vkAllocateCommandBuffers](#) (VkCommandPool commandPool, uint32\_t commandBufferCount, VkCommandBuffer \*pCommandBuffers, VkCommandBufferLevel level=VK\_COMMAND\_BUFFER\_LEVEL\_PRIMARY)
- void [vkDestroyImageView](#) (VkImageView imageView, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- void [vkDestroyFence](#) (VkFence fence, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- void [vkDestroySemaphore](#) (VkSemaphore semaphore, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- void [vkFreeCommandBuffers](#) (VkCommandPool commandPool, uint32\_t commandBufferCount, const VkCommandBuffer \*pCommandBuffers)
- void [vkDestroyCommandPool](#) (VkCommandPool commandPool, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- void [vkDestroySampler](#) (VkSampler sampler, const VkAllocationCallbacks \*pAllocator=[memory::instance\(\).alloc\(\)](#))
- void [vkUpdateDescriptorSets](#) (uint32\_t descriptorWriteCount, const VkWriteDescriptorSet \*pDescriptorWrites, uint32\_t descriptorCopyCount=0, const VkCopyDescriptorSet \*pDescriptorCopies=nullptr)
- template<std::size\_t SIZE>  
void [vkUpdateDescriptorSets](#) (std::array< VkWriteDescriptorSet, SIZE > const &descriptor\_writes)
- template<std::size\_t SIZE>  
void [vkUpdateDescriptorSets](#) (std::array< VkCopyDescriptorSet, SIZE > const &descriptor\_copies)
- template<std::size\_t WRITE\_SIZE, std::size\_t COPY\_SIZE>  
void [vkUpdateDescriptorSets](#) (std::array< VkWriteDescriptorSet, WRITE\_SIZE > const &descriptor\_writes, std::array< VkCopyDescriptorSet, COPY\_SIZE > const &descriptor\_copies)
- void [vkUpdateDescriptorSets](#) (std::initializer\_list< VkWriteDescriptorSet > descriptor\_writes)
- void [vkUpdateDescriptorSets](#) (std::initializer\_list< VkCopyDescriptorSet > descriptor\_copies)
- void [vkUpdateDescriptorSets](#) (std::initializer\_list< VkWriteDescriptorSet > descriptor\_writes, std::initializer\_list< VkCopyDescriptorSet > descriptor\_copies)

## Public Attributes

- VkDevice **vk\_device** = nullptr  
*Vulkan device.*
- VolkDeviceTable **table** = {}  
*Volk device table.*

## 4.27.1 Detailed Description

Device functions.

## 4.27.2 Member Function Documentation

### 4.27.2.1 vkAcquireNextImageKHR()

```
vk_result lava::device_table::vkAcquireNextImageKHR (
    VkSwapchainKHR swapchain,
    uint64_t timeout,
    VkSemaphore semaphore,
    VkFence fence,
    uint32_t * pImageIndex) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkAcquireNextImageKHR.html>

#### 4.27.2.2 vkAllocateCommandBuffers() [1/2]

```
vk_result lava::device_table::vkAllocateCommandBuffers (
    const VkCommandBufferAllocateInfo * pAllocateInfo,
    VkCommandBuffer * pCommandBuffers) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkAllocateCommandBuffers>

#### 4.27.2.3 vkAllocateCommandBuffers() [2/2]

```
vk_result lava::device_table::vkAllocateCommandBuffers (
    VkCommandPool commandPool,
    uint32_t commandBufferCount,
    VkCommandBuffer * pCommandBuffers,
    VkCommandBufferLevel level = VK_COMMAND_BUFFER_LEVEL_PRIMARY) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkAllocateCommandBuffers>

#### 4.27.2.4 vkCreateCommandPool() [1/3]

```
vk_result lava::device_table::vkCreateCommandPool (
    const VkCommandPoolCreateInfo * pCreateInfo,
    const VkAllocationCallbacks * pAllocator,
    VkCommandPool * pCommandPool) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateCommandPool>

#### 4.27.2.5 vkCreateCommandPool() [2/3]

```
vk_result lava::device_table::vkCreateCommandPool (
    const VkCommandPoolCreateInfo * pCreateInfo,
    VkCommandPool * pCommandPool) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateCommandPool>

#### 4.27.2.6 vkCreateCommandPool() [3/3]

```
vk_result lava::device_table::vkCreateCommandPool (
    index queue_family,
    VkCommandPool * pCommandPool) [inline]
```

See also

[https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵  
CreateCommandPool](https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateCommandPool)

#### 4.27.2.7 vkCreateFence() [1/2]

```
vk_result lava::device_table::vkCreateFence (
    const VkFenceCreateInfo * pCreateInfo,
    const VkAllocationCallbacks * pAllocator,
    VkFence * pFence) [inline]
```

See also

[https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵  
CreateFence](https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵<br/>CreateFence)

#### 4.27.2.8 vkCreateFence() [2/2]

```
vk_result lava::device_table::vkCreateFence (
    const VkFenceCreateInfo * pCreateInfo,
    VkFence * pFence) [inline]
```

See also

[https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵  
CreateFence](https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵<br/>CreateFence)

#### 4.27.2.9 vkCreateImageView() [1/2]

```
vk_result lava::device_table::vkCreateImageView (
    const VkImageViewCreateInfo * pCreateInfo,
    const VkAllocationCallbacks * pAllocator,
    VkImageView * pView) [inline]
```

See also

[https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵  
CreateImageView](https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vk↵<br/>CreateImageView)



#### 4.27.2.10 vkCreateImageView() [2/2]

```
vk_result lava::device_table::vkCreateImageView (  
    const VkImageViewCreateInfo * pCreateInfo,  
    VkImageView * pView) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateImageView>

#### 4.27.2.11 vkCreateSampler() [1/2]

```
vk_result lava::device_table::vkCreateSampler (  
    const VkSamplerCreateInfo * pCreateInfo,  
    const VkAllocationCallbacks * pAllocator,  
    VkSampler * pSampler) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateSampler>

#### 4.27.2.12 vkCreateSampler() [2/2]

```
vk_result lava::device_table::vkCreateSampler (  
    const VkSamplerCreateInfo * pCreateInfo,  
    VkSampler * pSampler) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateSampler>

#### 4.27.2.13 vkCreateSemaphore() [1/2]

```
vk_result lava::device_table::vkCreateSemaphore (  
    const VkSemaphoreCreateInfo * pCreateInfo,  
    const VkAllocationCallbacks * pAllocator,  
    VkSemaphore * pSemaphore) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateSemaphore>

#### 4.27.2.14 vkCreateSemaphore() [2/2]

```
vk_result lava::device_table::vkCreateSemaphore (
    const VkSemaphoreCreateInfo * pCreateInfo,
    VkSemaphore * pSemaphore) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateSemaphore>

#### 4.27.2.15 vkCreateShaderModule() [1/2]

```
vk_result lava::device_table::vkCreateShaderModule (
    const VkShaderModuleCreateInfo * pCreateInfo,
    const VkAllocationCallbacks * pAllocator,
    VkShaderModule * pShaderModule) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateShaderModule>

#### 4.27.2.16 vkCreateShaderModule() [2/2]

```
vk_result lava::device_table::vkCreateShaderModule (
    const VkShaderModuleCreateInfo * pCreateInfo,
    VkShaderModule * pShaderModule) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkCreateShaderModule>

#### 4.27.2.17 vkCreateSwapchainKHR() [1/2]

```
vk_result lava::device_table::vkCreateSwapchainKHR (
    const VkSwapchainCreateInfoKHR * pCreateInfo,
    const VkAllocationCallbacks * pAllocator,
    VkSwapchainKHR * pSwapchain) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkCreateSwapchainKHR.html>

#### 4.27.2.18 vkCreateSwapchainKHR() [2/2]

```
vk_result lava::device_table::vkCreateSwapchainKHR (
    const VkSwapchainCreateInfoKHR * pCreateInfo,
    VkSwapchainKHR * pSwapchain) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkCreateSwapchainKHR.html>

#### 4.27.2.19 vkDestroyCommandPool()

```
void lava::device_table::vkDestroyCommandPool (
    VkCommandPool commandPool,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkDestroyCommandPool>

#### 4.27.2.20 vkDestroyFence()

```
void lava::device_table::vkDestroyFence (
    VkFence fence,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkDestroyFence>

#### 4.27.2.21 vkDestroyImageView()

```
void lava::device_table::vkDestroyImageView (
    VkImageView imageView,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkDestroyImageView>

#### 4.27.2.22 vkDestroySampler()

```
void lava::device_table::vkDestroySampler (
    VkSampler sampler,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

##### See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkDestroySampler>

#### 4.27.2.23 vkDestroySemaphore()

```
void lava::device_table::vkDestroySemaphore (
    VkSemaphore semaphore,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

##### See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkDestroySemaphore>

#### 4.27.2.24 vkDestroySwapchainKHR()

```
void lava::device_table::vkDestroySwapchainKHR (
    VkSwapchainKHR swapchain,
    const VkAllocationCallbacks * pAllocator = memory::instance().alloc()) [inline]
```

##### See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkDestroySwapchainKHR.html>

#### 4.27.2.25 vkFreeCommandBuffers()

```
void lava::device_table::vkFreeCommandBuffers (
    VkCommandPool commandPool,
    uint32_t commandBufferCount,
    const VkCommandBuffer * pCommandBuffers) [inline]
```

##### See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkFreeCommandBuffers>

#### 4.27.2.26 vkGetSwapchainImagesKHR()

```
vk_result lava::device_table::vkGetSwapchainImagesKHR (
    VkSwapchainKHR swapchain,
    uint32_t * pSwapchainImageCount,
    VkImage * pSwapchainImages) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkGetSwapchainImagesKHR.html>

#### 4.27.2.27 vkQueuePresentKHR()

```
vk_result lava::device_table::vkQueuePresentKHR (
    VkQueue queue,
    const VkPresentInfoKHR * pPresentInfo) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/vkQueuePresentKHR.html>

#### 4.27.2.28 vkQueueSubmit()

```
vk_result lava::device_table::vkQueueSubmit (
    VkQueue queue,
    uint32_t submitCount,
    const VkSubmitInfo * pSubmits,
    VkFence fence) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkQueueSubmit>

#### 4.27.2.29 vkResetFences()

```
vk_result lava::device_table::vkResetFences (
    uint32_t fenceCount,
    const VkFence * pFences) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkResetFences>

**4.27.2.30 vkUpdateDescriptorSets() [1/7]**

```
template<std::size_t SIZE>
void lava::device_table::vkUpdateDescriptorSets (
    std::array< VkCopyDescriptorSet, SIZE > const & descriptor_copies) [inline]
```

**See also**

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

**4.27.2.31 vkUpdateDescriptorSets() [2/7]**

```
template<std::size_t SIZE>
void lava::device_table::vkUpdateDescriptorSets (
    std::array< VkWriteDescriptorSet, SIZE > const & descriptor_writes) [inline]
```

**See also**

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

**4.27.2.32 vkUpdateDescriptorSets() [3/7]**

```
template<std::size_t WRITE_SIZE, std::size_t COPY_SIZE>
void lava::device_table::vkUpdateDescriptorSets (
    std::array< VkWriteDescriptorSet, WRITE_SIZE > const & descriptor_writes,
    std::array< VkCopyDescriptorSet, COPY_SIZE > const & descriptor_copies) [inline]
```

**See also**

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

**4.27.2.33 vkUpdateDescriptorSets() [4/7]**

```
void lava::device_table::vkUpdateDescriptorSets (
    std::initializer_list< VkCopyDescriptorSet > descriptor_copies) [inline]
```

**See also**

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

#### 4.27.2.34 vkUpdateDescriptorSets() [5/7]

```
void lava::device_table::vkUpdateDescriptorSets (
    std::initializer_list< VkWriteDescriptorSet > descriptor_writes) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

#### 4.27.2.35 vkUpdateDescriptorSets() [6/7]

```
void lava::device_table::vkUpdateDescriptorSets (
    std::initializer_list< VkWriteDescriptorSet > descriptor_writes,
    std::initializer_list< VkCopyDescriptorSet > descriptor_copies) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

#### 4.27.2.36 vkUpdateDescriptorSets() [7/7]

```
void lava::device_table::vkUpdateDescriptorSets (
    uint32_t descriptorWriteCount,
    const VkWriteDescriptorSet * pDescriptorWrites,
    uint32_t descriptorCopyCount = 0,
    const VkCopyDescriptorSet * pDescriptorCopies = nullptr) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkUpdateDescriptorSets>

#### 4.27.2.37 vkWaitForFences()

```
vk_result lava::device_table::vkWaitForFences (
    uint32_t fenceCount,
    const VkFence * pFences,
    VkBool32 waitAll,
    uint64_t timeout) [inline]
```

See also

<https://www.khronos.org/registry/vulkan/specs/1.3/html/vkspec.html#vkWaitForFences>

The documentation for this struct was generated from the following file:

- liblava/base/[device\\_table.hpp](#)

## 4.28 lava::driver Struct Reference

Stage driver.

```
#include <driver.hpp>
```

### Classes

- struct [result](#)  
*Driver result.*

### Public Types

- enum [error](#) { [stages\\_empty](#) = -1 , [stage\\_not\\_found](#) = -2 , [undef\\_run](#) = -3 }  
*Driver error codes.*
- using [run\\_func](#) = std::function<[result](#)(argh::parser)>  
*Run function.*

### Public Member Functions

- void [add\\_stage](#) ([stage](#) \*[stage](#))  
*Add a stage.*
- [stage::map](#) const & [get\\_stages](#) () const  
*Get all stages.*
- [i32](#) [run](#) (argh::parser [cmd\\_line](#)={})  
*Run the driver.*

### Static Public Member Functions

- static [driver](#) & [instance](#) ()  
*Get driver instance.*

### Public Attributes

- [run\\_func](#) [on\\_run](#)  
*Called if no stage has been selected.*

### 4.28.1 Detailed Description

Stage driver.

### 4.28.2 Member Function Documentation

#### 4.28.2.1 add\_stage()

```
void lava::driver::add_stage (  
    stage * stage) [inline]
```

Add a stage.



## Parameters

<i>stage</i>	Stage to add
--------------	--------------

### 4.28.2.2 get\_stages()

```
stage::map const & lava::driver::get_stages () const [inline]
```

Get all stages.

## Returns

stage::map const& Map of stages

### 4.28.2.3 instance()

```
static driver & lava::driver::instance () [inline], [static]
```

Get driver instance.

## Returns

driver& Stage driver

### 4.28.2.4 run()

```
i32 lava::driver::run (  
    argh::parser cmd_line = {})
```

Run the driver.

## Parameters

<i>cmd_line</i>	Command line arguments
-----------------	------------------------

## Returns

i32 Result code

The documentation for this struct was generated from the following file:

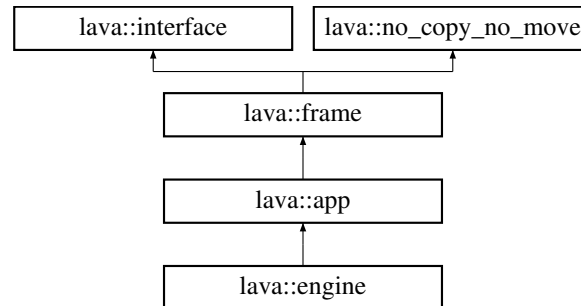
- liblava/frame/[driver.hpp](#)

## 4.29 lava::engine Struct Reference

Engine.

```
#include <engine.hpp>
```

Inheritance diagram for lava::engine:



### Public Types

- using **ptr** = [engine\\*](#)
- using **hud\_menu\_func** = std::function<void()>  
*Hud menu function.*

### Public Types inherited from [lava::app](#)

- using **update\_func** = std::function<bool([delta](#))>  
*Update function.*
- using **create\_func** = std::function<bool()>  
*Create function.*
- using **destroy\_func** = std::function<void()>  
*Destroy function.*
- using **process\_func** = std::function<void(VkCommandBuffer, [index](#))>  
*Process function.*
- using **setup\_func** = std::function<bool()>  
*Set up function.*

### Public Types inherited from [lava::frame](#)

- using **s\_ptr** = std::shared\_ptr<[frame](#)>  
*Shared pointer to framework.*
- using **result** = [i32](#)  
*Framework result.*
- using **run\_func** = std::function<bool([id::ref](#))>  
*Run function.*
- using **run\_func\_ref** = [run\\_func](#) const&  
*Reference to run function.*
- using **run\_end\_func** = std::function<void()>  
*Run end function.*
- using **run\_end\_func\_ref** = [run\\_end\\_func](#) const&  
*Reference to run end function.*
- using **run\_once\_func** = std::function<bool()>  
*Run once function.*
- using **run\_once\_func\_ref** = [run\\_once\\_func](#) const&  
*Reference to run once function.*

## Public Member Functions

- bool [setup](#) () override  
*Set up the engine.*
- [app](#) ([frame\\_env::ref](#) env)  
*App constructors.*
- [app](#) ([name](#) name, [argh::parser](#) cmd\_line={})  
*App constructors.*

## Public Member Functions inherited from [lava::app](#)

- [app](#) ([frame\\_env::ref](#) env)  
*Construct a new app.*
- [app](#) ([name](#) name, [argh::parser](#) cmd\_line={})  
*Construct a new app.*
- bool [v\\_sync](#) () const  
*V-Sync setting.*
- bool [triple\\_buffer](#) () const  
*Triple buffering setting.*
- [ui32](#) [fps\\_cap](#) () const  
*Frames per second cap setting.*
- [ui32](#) [get\\_frame\\_counter](#) () const  
*Get the frame counter.*
- [string](#) [get\\_fps\\_info](#) () const  
*Get frames per second info.*
- void [draw\\_about](#) ([about\\_info\\_setting](#) setting=[about\\_info\\_setting::all](#)()) const  
*Draw about information.*
- [id::ref](#) [block\\_cmd](#) () const  
*Get id of the block command.*
- [string](#) [screenshot](#) ()  
*Take screenshot and save it to file.*
- void [switch\\_config](#) ([string\\_ref](#) config\_name)

## Public Member Functions inherited from [lava::frame](#)

- [frame](#) ([argh::parser](#) cmd\_line)  
*Construct a new framework.*
- [frame](#) ([frame\\_env](#) env)  
*Construct a new framework.*
- [~frame](#) () override  
*Destroy the framework.*
- bool [ready](#) () const  
*Check if framework is ready.*
- [result](#) [run](#) ()  
*Run the framework.*
- bool [shut\\_down](#) ()  
*Shut down the framework.*
- [id](#) [add\\_run](#) ([run\\_func\\_ref](#) func)  
*Add run to framework.*
- [id](#) [add\\_run\\_end](#) ([run\\_end\\_func\\_ref](#) func)

- Add run end to framework.*
  - void [add\\_run\\_once](#) ([run\\_once\\_func\\_ref](#) func)
- Add run once to framework.*
  - bool [remove](#) ([id::ref](#) func\_id)
- Remove a function from framework.*
  - [ms\\_get\\_running\\_time](#) () const
- Get the running time.*
  - [r64\\_get\\_running\\_time\\_sec](#) () const
- Get the running time in seconds.*
  - [cmd\\_line\\_get\\_cmd\\_line](#) () const
- Get the command line arguments.*
  - [frame\\_env::ref\\_get\\_env](#) () const
- Get the framework environment.*
  - [name\\_get\\_name](#) () const
- Get the name of application.*
  - bool [waiting\\_for\\_events](#) () const
- Check if framework is waiting for events.*
  - void [set\\_wait\\_for\\_events](#) (bool value=true)
- Set wait for events in framework.*

## Public Member Functions inherited from [lava::interface](#)

- virtual [~interface](#) ()=default
- Destroy the interface.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default
- Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete
- No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete
- No move.*

## Public Attributes

- [lava::props](#) **props**
- Props.*
- [lava::producer](#) **producer**
- Producer.*
- [hud\\_menu\\_func](#) **on\_menu**
- Function called on hud menu.*
- bool **hud\_active** = false
- Hud active state.*

## Public Attributes inherited from [lava::app](#)

- **bool headless** = false  
*Headless mode: no window, no input, no camera, no renderer, no block, no target, no shading, no gamepad. Enable it before calling the setup method.*
- [lava::window](#) **window**  
*Main window.*
- [lava::input](#) **input**  
*Window input.*
- [lava::imgui](#) **imgui**  
*ImGui handling.*
- [imgui::config](#) **imgui\_config**  
*ImGui configuration.*
- [tooltip\\_list](#) **tooltips**  
*Tooltip list.*
- [lava::device::ptr](#) **device** = nullptr  
*Vulkan device.*
- [lava::camera](#) **camera**  
*Main camera.*
- [gamepad](#) **pad**  
*Gamepad.*
- [lava::staging](#) **staging**  
*Texture staging.*
- [lava::block](#) **block**  
*Basic block.*
- [lava::renderer](#) **renderer**  
*Plain renderer.*
- [forward\\_shading](#) **shading**  
*Forward shading.*
- [render\\_target::s\\_ptr](#) **target**  
*Render target.*
- [file\\_system](#) **fs**  
*File system.*
- [VkPipelineCache](#) **pipeline\_cache** = nullptr  
*Pipeline cache.*
- [update\\_func](#) **on\_update**  
*Function called on application update.*
- [create\\_func](#) **on\_create**  
*Function called on application create.*
- [destroy\\_func](#) **on\_destroy**  
*Function called on application destroy.*
- [app\\_config](#) **config**  
*Application configuration.*
- [json\\_file](#) **config\_file**  
*Configuration file.*
- [process\\_func](#) **on\_process**  
*Function called on application process.*
- [setup\\_func](#) **on\_setup**  
*Function called on application setup.*

## Public Attributes inherited from [lava::frame](#)

- [lava::run\\_time](#) **run\_time**  
*Run time.*
- [lava::platform](#) **platform**  
*Stage platform.*
- [message\\_dispatcher](#) **telegraph**  
*Message dispatcher.*

### 4.29.1 Detailed Description

Engine.

### 4.29.2 Member Function Documentation

#### 4.29.2.1 setup()

```
bool lava::engine::setup () [override], [virtual]
```

Set up the engine.

#### Returns

Setup was successful or failed

Reimplemented from [lava::app](#).

The documentation for this struct was generated from the following file:

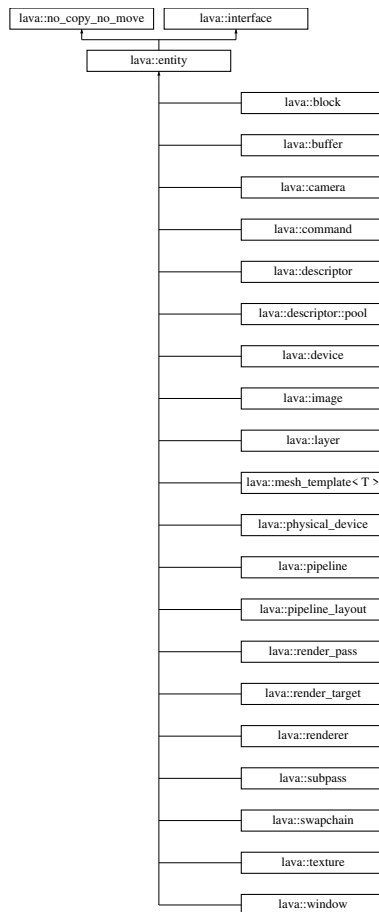
- [liblava/engine/engine.hpp](#)

## 4.30 lava::entity Struct Reference

Entity.

```
#include <id.hpp>
```

Inheritance diagram for `lava::entity`:



### Public Member Functions

- **entity ()**  
*Construct a new entity.*
- **id::ref get\_id () const**  
*Get the id of entity.*

### Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move ()=default**  
*Construct a new object.*
- **no\_copy\_no\_move (no\_copy\_no\_move const &)=delete**  
*No copy.*
- **void operator= (no\_copy\_no\_move const &)=delete**  
*No move.*

### Public Member Functions inherited from [lava::interface](#)

- **virtual ~interface ()=default**  
*Destroy the interface.*

### 4.30.1 Detailed Description

Entity.

### 4.30.2 Member Function Documentation

#### 4.30.2.1 `get_id()`

```
id::ref lava::entity::get_id () const [inline]
```

Get the id of entity.

Returns

`id::ref` Entity id

The documentation for this struct was generated from the following file:

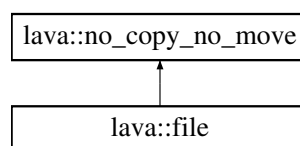
- `liblava/core/id.hpp`

## 4.31 `lava::file` Struct Reference

File.

```
#include <file.hpp>
```

Inheritance diagram for `lava::file`:



### Public Types

- using `ref = file` const&  
*Reference to file.*



### Public Member Functions

- `file` (`string_ref` path="", `file_mode` mode=`file_mode::read`)  
*Construct a new file.*
- `~file` ()  
*Destroy the file.*
- `bool open` (`string_ref` path, `file_mode` mode=`file_mode::read`)  
*Open the file.*
- `void close` ()  
*Close the file.*
- `bool opened` () const  
*Check if the file is opened.*
- `i64 get_size` () const  
*Get the size of the file.*
- `i64 read` (`data::ptr` data)  
*Read data from file.*
- `i64 read` (`data::ptr` data, `ui64` size)  
*Read data from file (limited size)*
- `i64 write` (`data::c_ptr` data, `ui64` size)  
*Write data to file.*
- `i64 seek` (`ui64` position)  
*Seek to position in the file.*
- `i64 tell` () const  
*Get the current position in the file.*
- `bool writable` () const  
*Check if the file is in write mode.*
- `file_type get_type` () const  
*Get the file type.*
- `string_ref get_path` () const  
*Get the path of the file.*

### Public Member Functions inherited from `lava::no_copy_no_move`

- `no_copy_no_move` ()=default  
*Construct a new object.*
- `no_copy_no_move` (`no_copy_no_move` const &)=delete  
*No copy.*
- `void operator=` (`no_copy_no_move` const &)=delete  
*No move.*

#### 4.31.1 Detailed Description

File.

#### 4.31.2 Constructor & Destructor Documentation

##### 4.31.2.1 `file()`

```

lava::file::file (
    string_ref path = "",
    file_mode mode = file_mode::read) [explicit]

```

Construct a new file.

**Parameters**

<i>path</i>	Name of file
<i>mode</i>	File mode

### 4.31.3 Member Function Documentation

#### 4.31.3.1 `get_path()`

```
string_ref lava::file::get_path () const [inline]
```

Get the path of the file.

**Returns**

name File path

#### 4.31.3.2 `get_size()`

```
i64 lava::file::get_size () const
```

Get the size of the file.

**Returns**

i64 File size

#### 4.31.3.3 `get_type()`

```
file_type lava::file::get_type () const [inline]
```

Get the file type.

**Returns**

file\_type Type of file

#### 4.31.3.4 `open()`

```
bool lava::file::open (  
    string_ref path,  
    file_mode mode = file_mode::read)
```

Open the file.

## Parameters

<i>path</i>	Name of file
<i>mode</i>	File mode

## Returns

Open was successful or failed

**4.31.3.5 opened()**

```
bool lava::file::opened () const
```

Check if the file is opened.

## Returns

File is opened or not

**4.31.3.6 read() [1/2]**

```
i64 lava::file::read (  
    data::ptr data) [inline]
```

Read data from file.

## Parameters

<i>data</i>	Data to read
-------------	--------------

## Returns

i64 File size

**4.31.3.7 read() [2/2]**

```
i64 lava::file::read (  
    data::ptr data,  
    ui64 size)
```

Read data from file (limited size)

## Parameters

<i>data</i>	Data to read
<i>size</i>	File size

## Returns

i64 File size

**4.31.3.8 seek()**

```
i64 lava::file::seek (  
    ui64 position)
```

Seek to position in the file.

**Parameters**

<i>position</i>	Position to seek to
-----------------	---------------------

**Returns**

i64 Current position

**4.31.3.9 tell()**

```
i64 lava::file::tell () const
```

Get the current position in the file.

**Returns**

i64 Current position

**4.31.3.10 writable()**

```
bool lava::file::writable () const [inline]
```

Check if the file is in write mode.

**Returns**

File is writable or only readable

**4.31.3.11 write()**

```
i64 lava::file::write (  
    data::c_ptr data,  
    ui64 size)
```

Write data to file.

**Parameters**

<i>data</i>	Data to write
<i>size</i>	File size

**Returns**

i64 File size

The documentation for this struct was generated from the following file:

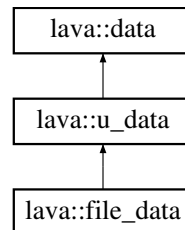
- [liblava/file/file.hpp](#)

## 4.32 `lava::file_data` Struct Reference

File data.

```
#include <file_utils.hpp>
```

Inheritance diagram for `lava::file_data`:



### Public Types

- using **ref** = `file_data` const&  
*Reference to file data.*

### Public Types inherited from `lava::u_data`

- using **ref** = `u_data` const&  
*Reference to unique data wrapper.*

### Public Types inherited from `lava::data`

- enum class `mode` : index { **alloc** = 0 , **no\_alloc** }
- *Data modes.*
- using **ref** = `data` const&  
*Reference to data wrapper.*
- using **ptr** = char\*
- *Data pointer.*
- using **c\_ptr** = char const\*
- *Const data pointer.*

### Public Member Functions

- `file_data` (string\_ref filename)  
*Construct a new file data.*
- `u_data` (size\_t length=0, `data::mode` mode=`data::mode::alloc`)  
*Unique data constructors.*
- `u_data` (`data::ref` data)  
*Unique data constructors.*

## Public Member Functions inherited from [lava::u\\_data](#)

- [u\\_data](#) ([size\\_t](#) length=0, [data::mode](#) mode=[data::mode::alloc](#))  
*Construct a new unique data.*
- [u\\_data](#) ([data::ref](#) data)  
*Construct a new unique data from another data.*
- [~u\\_data](#) ()  
*Destroy the unique data.*

## Public Member Functions inherited from [lava::data](#)

- [data](#) ()=default  
*Construct a new data.*
- [data](#) (auto \*[addr](#), [size\\_t](#) size)  
*Construct a new data.*
- bool [set](#) ([size\\_t](#) length, [mode](#) mode=[mode::alloc](#))  
*Set and allocate data by length.*
- bool [allocate](#) ()  
*Allocate data.*
- void [deallocate](#) ()  
*Deallocate data.*
- [ptr](#) [end](#) () const  
*Pointer to end of data.*

## Public Attributes

- [string](#) [filename](#)  
*Name of file.*

## Public Attributes inherited from [lava::data](#)

- [ptr](#) [addr](#) = nullptr  
*Pointer address.*
- [size\\_t](#) [size](#) = 0  
*Size of data.*
- [size\\_t](#) [alignment](#) = 0  
*Data alignment.*

## Additional Inherited Members

## Static Public Member Functions inherited from [lava::data](#)

- static [ptr](#) [as\\_ptr](#) (auto \*value)  
*Cast to data pointer.*
- static [c\\_ptr](#) [as\\_c\\_ptr](#) (auto \*value)  
*Cast to const data pointer.*

### 4.32.1 Detailed Description

File data.

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 file\_data()

```
lava::file_data::file_data (
    string_ref filename) [inline], [explicit]
```

Construct a new file data.

Parameters

<i>filename</i>	Name of file
-----------------	--------------

The documentation for this struct was generated from the following file:

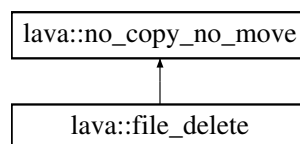
- [liblava/file/file\\_utils.hpp](#)

## 4.33 lava::file\_delete Struct Reference

File delete guard.

```
#include <file_utils.hpp>
```

Inheritance diagram for lava::file\_delete:



### Public Member Functions

- [file\\_delete](#) (string filename="")  
*Construct a new file delete.*
- [~file\\_delete](#) ()  
*Destroy the file delete.*

### Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) (no\_copy\_no\_move const &)=delete  
*No copy.*
- void [operator=](#) (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Attributes

- `string filename`  
*Name of file.*
- `bool active = true`  
*Active state.*

### 4.33.1 Detailed Description

File delete guard.

### 4.33.2 Constructor & Destructor Documentation

#### 4.33.2.1 `file_delete()`

```
lava::file_delete::file_delete (  
    string filename = "") [inline], [explicit]
```

Construct a new file delete.

#### Parameters

<code>filename</code>	Name of file
-----------------------	--------------

The documentation for this struct was generated from the following file:

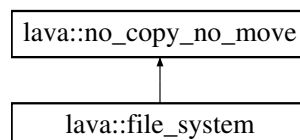
- [liblava/file/file\\_utils.hpp](#)

## 4.34 `lava::file_system` Struct Reference

File system.

```
#include <file_system.hpp>
```

Inheritance diagram for `lava::file_system`:





## Public Member Functions

- [sem\\_version](#) [get\\_version](#) ()  
*Get the version.*
- [string](#) [get\\_base\\_dir](#) ()  
*Get the base directory.*
- [string](#) [get\\_full\\_base\\_dir](#) ([string\\_ref](#) path)  
*Get the path relative to base directory.*
- [string](#) [get\\_pref\\_dir](#) ()  
*Get the preferences directory.*
- [string](#) [get\\_res\\_dir](#) ()  
*Get the resource directory.*
- [bool](#) [mount](#) ([string\\_ref](#) path)  
*Mount path.*
- [bool](#) [mount\\_base](#) ([string\\_ref](#) base\_dir\_path)  
*Mount base directory path.*
- [bool](#) [exists](#) ([string\\_ref](#) file)  
*Check if file exists.*
- [string](#) [get\\_real\\_dir](#) ([string\\_ref](#) file)  
*Get the real directory of file.*
- [string\\_list](#) [enumerate\\_files](#) ([string\\_ref](#) path)  
*Enumerate files in directory.*
- [bool](#) [initialize](#) ([string\\_ref](#) argv\_0, [string\\_ref](#) org, [string\\_ref](#) app, [string\\_ref](#) ext)  
*Initialize the file system.*
- [void](#) [terminate](#) ()  
*Terminate the file system.*
- [string\\_list](#) [mount\\_res](#) ()  
*Mount resource directories and files.*
- [bool](#) [create\\_folder](#) ([string\\_ref](#) name="data")  
*Create a folder in the preferences directory.*
- [void](#) [clean\\_pref\\_dir](#) ()  
*Clean preferences directory.*
- [string\\_ref](#) [get\\_org](#) () const  
*Get the organization name.*
- [string\\_ref](#) [get\\_app](#) () const  
*Get the application name.*
- [string\\_ref](#) [get\\_ext](#) () const  
*Get the extension name.*
- [bool](#) [ready](#) () const  
*Check if file system is ready.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- [void](#) [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

### 4.34.1 Detailed Description

File system.

### 4.34.2 Member Function Documentation

#### 4.34.2.1 create\_folder()

```
bool lava::file_system::create_folder (  
    string_ref name = "data")
```

Create a folder in the preferences directory.

##### Parameters

<i>name</i>	Name of folder (default: data)
-------------	--------------------------------

##### Returns

Folder created or not

#### 4.34.2.2 enumerate\_files()

```
string_list lava::file_system::enumerate_files (  
    string_ref path)
```

Enumerate files in directory.

##### Parameters

<i>path</i>	Target directory
-------------	------------------

##### Returns

string\_list List of files

#### 4.34.2.3 exists()

```
bool lava::file_system::exists (  
    string_ref file)
```

Check if file exists.

##### Parameters

<i>file</i>	File to check
-------------	---------------

##### Returns

File exists or not found

#### 4.34.2.4 get\_app()

```
string_ref lava::file_system::get_app () const [inline]
```

Get the application name.

##### Returns

string\_ref Name of application

#### 4.34.2.5 get\_base\_dir()

```
string lava::file_system::get_base_dir ()
```

Get the base directory.

##### Returns

string Base directory

#### 4.34.2.6 get\_ext()

```
string_ref lava::file_system::get_ext () const [inline]
```

Get the extension name.

##### Returns

string\_ref Name of extension

#### 4.34.2.7 get\_full\_base\_dir()

```
string lava::file_system::get_full_base_dir (  
    string_ref path)
```

Get the path relative to base directory.

##### Parameters

<i>path</i>	Path to add to base directory
-------------	-------------------------------

##### Returns

string Relative base directory path

#### 4.34.2.8 get\_org()

```
string_ref lava::file_system::get_org () const [inline]
```

Get the organization name.

##### Returns

string\_ref Name of organization

#### 4.34.2.9 get\_pref\_dir()

```
string lava::file_system::get_pref_dir ()
```

Get the preferences directory.

##### Returns

string Preferences directory

#### 4.34.2.10 get\_real\_dir()

```
string lava::file_system::get_real_dir (  
    string_ref file)
```

Get the real directory of file.

##### Parameters

<i>file</i>	Target file
-------------	-------------

##### Returns

string Real directory of file

#### 4.34.2.11 get\_res\_dir()

```
string lava::file_system::get_res_dir ()
```

Get the resource directory.

##### Returns

string Resource directory

#### 4.34.2.12 get\_version()

```
sem_version lava::file_system::get_version ()
```

Get the version.

##### Returns

sem\_version Semantic version

#### 4.34.2.13 initialize()

```
bool lava::file_system::initialize (  
    string_ref argv_0,  
    string_ref org,  
    string_ref app,  
    string_ref ext)
```

Initialize the file system.

##### Parameters

<i>argv_0</i>	First command line argument
<i>org</i>	Organization name
<i>app</i>	Application name
<i>ext</i>	Extension name

##### Returns

Initialize was successful or failed

#### 4.34.2.14 mount()

```
bool lava::file_system::mount (  
    string_ref path)
```

Mount path.

##### Parameters

<i>path</i>	Path to mount
-------------	---------------

##### Returns

Mount was successful or failed

#### 4.34.2.15 mount\_base()

```
bool lava::file_system::mount_base (  
    string_ref base_dir_path)
```

Mount base directory path.

## Parameters

<code>base_dir_path</code>	Base directory path
----------------------------	---------------------

## Returns

Mount was successful or failed

**4.34.2.16 mount\_res()**

```
string_list lava::file_system::mount_res ()
```

Mount resource directories and files.

## Returns

string\_list List of mounted resources

**4.34.2.17 ready()**

```
bool lava::file_system::ready () const [inline]
```

Check if file system is ready.

## Returns

File system is ready or not

The documentation for this struct was generated from the following file:

- [liblava/file/file\\_system.hpp](#)

**4.35 lava::imgui::font Struct Reference**

ImGui font settings.

```
#include <imgui.hpp>
```

**Public Types**

- using **ref** = [font](#) const&  
*Const font reference.*

### Public Attributes

- **string** **file**  
*Font file.*
- **r32** **size** = 21.f  
*Font size.*
- **string** **icon\_file**  
*Font icon file.*
- **r32** **icon\_size** = 21.f  
*Font icon size.*
- **ui16** **icon\_range\_begin** = 0  
*Font range begin.*
- **ui16** **icon\_range\_end** = 0  
*Font range end.*

### 4.35.1 Detailed Description

ImGui font settings.

The documentation for this struct was generated from the following file:

- [liblava/app/imgui.hpp](#)

## 4.36 lava::forward\_shading Struct Reference

Forward shading.

```
#include <forward_shading.hpp>
```

### Public Member Functions

- **forward\_shading** ()=default  
*Construct a new forward shading.*
- **~forward\_shading** ()  
*Destroy the forward shading.*
- bool **create** (**render\_target::s\_ptr** target)  
*Create a forward shading for a render target.*
- void **destroy** ()  
*Destroy the forward shading.*
- **render\_pass::s\_ptr** **get\_pass** () const  
*Get the render pass.*
- **VkRenderPass** **get\_vk\_pass** () const  
*Get the Vulkan render pass.*
- **image::s\_ptr** **get\_depth\_stencil** () const  
*Get the depth stencil image.*

### 4.36.1 Detailed Description

Forward shading.

### 4.36.2 Member Function Documentation

#### 4.36.2.1 create()

```
bool lava::forward_shading::create (  
    render_target::s_ptr target)
```

Create a forward shading for a render target.

##### Parameters

<i>target</i>	Render target
---------------	---------------

##### Returns

Create was successful or failed

#### 4.36.2.2 get\_depth\_stencil()

```
image::s_ptr lava::forward_shading::get_depth_stencil () const [inline]
```

Get the depth stencil image.

##### Returns

[image::s\\_ptr](#) Depth stencil Image

#### 4.36.2.3 get\_pass()

```
render_pass::s_ptr lava::forward_shading::get_pass () const [inline]
```

Get the render pass.

##### Returns

[render\\_pass::s\\_ptr](#) Render pass



## 4.36.2.4 get\_vk\_pass()

```
VkRenderPass lava::forward_shading::get_vk_pass () const [inline]
```

Get the Vulkan render pass.

## Returns

VkRenderPass Vulkan Render pass

The documentation for this struct was generated from the following file:

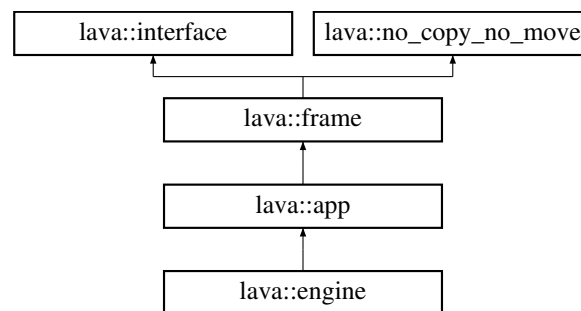
- liblava/app/[forward\\_shading.hpp](#)

## 4.37 lava::frame Struct Reference

Framework.

```
#include <frame.hpp>
```

Inheritance diagram for lava::frame:



## Public Types

- using **s\_ptr** = std::shared\_ptr<[frame](#)>  
*Shared pointer to framework.*
- using **result** = [i32](#)  
*Framework result.*
- using **run\_func** = std::function<bool([id::ref](#))>  
*Run function.*
- using **run\_func\_ref** = [run\\_func](#) const&  
*Reference to run function.*
- using **run\_end\_func** = std::function<void()>  
*Run end function.*
- using **run\_end\_func\_ref** = [run\\_end\\_func](#) const&  
*Reference to run end function.*
- using **run\_once\_func** = std::function<bool()>  
*Run once function.*
- using **run\_once\_func\_ref** = [run\\_once\\_func](#) const&  
*Reference to run once function.*

## Public Member Functions

- [frame](#) ([argh::parser](#) [cmd\\_line](#))  
*Construct a new framework.*
- [frame](#) ([frame\\_env](#) env)  
*Construct a new framework.*
- [~frame](#) () override  
*Destroy the framework.*
- bool [ready](#) () const  
*Check if framework is ready.*
- [result](#) [run](#) ()  
*Run the framework.*
- bool [shut\\_down](#) ()  
*Shut down the framework.*
- [id](#) [add\\_run](#) ([run\\_func\\_ref](#) func)  
*Add run to framework.*
- [id](#) [add\\_run\\_end](#) ([run\\_end\\_func\\_ref](#) func)  
*Add run end to framework.*
- void [add\\_run\\_once](#) ([run\\_once\\_func\\_ref](#) func)  
*Add run once to framework.*
- bool [remove](#) ([id::ref](#) func\_id)  
*Remove a function from framework.*
- [ms](#) [get\\_running\\_time](#) () const  
*Get the running time.*
- [r64](#) [get\\_running\\_time\\_sec](#) () const  
*Get the running time in seconds.*
- [cmd\\_line](#) [get\\_cmd\\_line](#) () const  
*Get the command line arguments.*
- [frame\\_env::ref](#) [get\\_env](#) () const  
*Get the framework environment.*
- [name](#) [get\\_name](#) () const  
*Get the name of application.*
- bool [waiting\\_for\\_events](#) () const  
*Check if framework is waiting for events.*
- void [set\\_wait\\_for\\_events](#) (bool value=true)  
*Set wait for events in framework.*

## Public Member Functions inherited from [lava::interface](#)

- virtual [~interface](#) ()=default  
*Destroy the interface.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Attributes

- [lava::run\\_time](#) **run\_time**  
*Run time.*
- [lava::platform](#) **platform**  
*Stage platform.*
- [message\\_dispatcher](#) **telegraph**  
*Message dispatcher.*

## 4.37.1 Detailed Description

Framework.

## 4.37.2 Constructor & Destructor Documentation

### 4.37.2.1 frame() [1/2]

```
lava::frame::frame (
    argh::parser cmd_line) [explicit]
```

Construct a new framework.

#### Parameters

<i>cmd_line</i>	Command line arguments
-----------------	------------------------

### 4.37.2.2 frame() [2/2]

```
lava::frame::frame (
    frame_env env) [explicit]
```

Construct a new framework.

#### Parameters

<i>env</i>	Framework environment
------------	-----------------------

## 4.37.3 Member Function Documentation

### 4.37.3.1 add\_run()

```
id lava::frame::add_run (
    run_func_ref func)
```

Add run to framework.

**Parameters**

<i>func</i>	Run function
-------------	--------------

**Returns**

id Id of function

**4.37.3.2 add\_run\_end()**

```
id lava::frame::add_run_end (  
    run_end_func_ref func)
```

Add run end to framework.

**Parameters**

<i>func</i>	Run end function
-------------	------------------

**Returns**

id Id of function

**4.37.3.3 add\_run\_once()**

```
void lava::frame::add_run_once (  
    run_once_func_ref func) [inline]
```

Add run once to framework.

**Parameters**

<i>func</i>	Run once function
-------------	-------------------

**4.37.3.4 get\_cmd\_line()**

```
cmd_line lava::frame::get_cmd_line () const [inline]
```

Get the command line arguments.

**Returns**

cmd\_line Command line arguments

#### 4.37.3.5 get\_env()

```
frame_env::ref lava::frame::get_env () const [inline]
```

Get the framework environment.

Returns

`frame_env::ref` Framework environment

#### 4.37.3.6 get\_name()

```
name lava::frame::get_name () const [inline]
```

Get the name of application.

Returns

`name` Name of application

#### 4.37.3.7 get\_running\_time()

```
ms lava::frame::get_running_time () const [inline]
```

Get the running time.

Returns

`ms` Time since start of framework

#### 4.37.3.8 get\_running\_time\_sec()

```
r64 lava::frame::get_running_time_sec () const [inline]
```

Get the running time in seconds.

Returns

`r64` Time since start of framework

#### 4.37.3.9 ready()

```
bool lava::frame::ready () const [inline]
```

Check if framework is ready.

Returns

Framework is ready or not

#### 4.37.3.10 remove()

```
bool lava::frame::remove (  
    id::ref func_id)
```

Remove a function from framework.

## Parameters

<i>func</i> ↔	Id of function
<i>_id</i>	

## Returns

Remove was successful or failed

**4.37.3.11 run()**

```
result lava::frame::run ()
```

Run the framework.

## Returns

result Run result

**4.37.3.12 set\_wait\_for\_events()**

```
void lava::frame::set_wait_for_events (
    bool value = true) [inline]
```

Set wait for events in framework.

## Parameters

<i>value</i>	Wait for events state
--------------	-----------------------

**4.37.3.13 shut\_down()**

```
bool lava::frame::shut_down ()
```

Shut down the framework.

## Returns

Shut down was successful or failed

**4.37.3.14 waiting\_for\_events()**

```
bool lava::frame::waiting_for_events () const [inline]
```

Check if framework is waiting for events.

## Returns

Framework waits for events or not

The documentation for this struct was generated from the following file:

- liblava/frame/[frame.hpp](#)

## 4.38 `lava::frame_env` Struct Reference

Framework environment.

```
#include <frame.hpp>
```

### Public Types

- using `ref` = `frame_env` const&  
*Reference to frame environment.*

### Public Member Functions

- `frame_env` ()  
*Construct a new frame environment.*
- `frame_env` (`name` app\_name, `argh::parser` `cmd_line`)  
*Construct a new frame environment.*
- void `set_default` ()  
*Set default settings.*

### Public Attributes

- `argh::parser` `cmd_line`  
*Command line arguments.*
- `log::config` `log`  
*Logging configuration.*
- `instance_info` `info`  
*Instance information.*
- `instance::create_param` `param`  
*Instance create parameters.*
- `instance::debug_config` `debug`  
*Intance debug configuration.*
- `ui32` `telegraph_thread_count` = 4  
*Message dispatcher threads.*

### 4.38.1 Detailed Description

Framework environment.

### 4.38.2 Constructor & Destructor Documentation

#### 4.38.2.1 `frame_env`()

```
lava::frame_env::frame_env (
    name app_name,
    argh::parser cmd_line) [inline], [explicit]
```

Construct a new frame environment.

## Parameters

<i>app_name</i>	Name of application
<i>cmd_line</i>	Command line arguments

The documentation for this struct was generated from the following file:

- [liblava/frame/frame.hpp](#)

## 4.39 lava::gamepad Struct Reference

Gamepad.

```
#include <gamepad.hpp>
```

### Public Types

- using **list** = std::vector<[gamepad](#)>  
*List of gamepads.*
- using **ref** = [gamepad](#) const&  
*Reference to gamepad.*

### Public Member Functions

- [gamepad](#) ([gamepad\\_id\\_ref](#) pad\_id=gamepad\_id::\_1)  
*Construct a new gamepad.*
- bool [ready](#) () const  
*Check if gamepad is active.*
- bool [update](#) ()  
*Update gamepad.*
- bool [pressed](#) ([gamepad\\_button\\_ref](#) button) const  
*Check if gamepad button is pressed.*
- [r32 value](#) ([gamepad\\_axis\\_ref](#) axis) const  
*Get value of axis.*
- [gamepad\\_id\\_ref](#) [get\\_pad\\_id](#) () const  
*Get the gamepad id.*
- [ui32](#) [get\\_id](#) () const  
*Get the gamepad id as integer.*
- [name](#) [get\\_name](#) () const  
*Get the name.*

### 4.39.1 Detailed Description

Gamepad.

### 4.39.2 Constructor & Destructor Documentation

#### 4.39.2.1 gamepad()

```
lava::gamepad::gamepad (  
    gamepad\_id\_ref pad_id = gamepad_id::_1) [explicit]
```

Construct a new gamepad.



## Parameters

<i>pad</i> ↔ _id	Gamepad id
---------------------	------------

### 4.39.3 Member Function Documentation

#### 4.39.3.1 get\_id()

```
ui32 lava::gamepad::get_id () const [inline]
```

Get the gamepad id as integer.

## Returns

ui32 Integer gamepad id

#### 4.39.3.2 get\_name()

```
name lava::gamepad::get_name () const
```

Get the name.

## Returns

name Name of gamepad

#### 4.39.3.3 get\_pad\_id()

```
gamepad_id_ref lava::gamepad::get_pad_id () const [inline]
```

Get the gamepad id.

## Returns

gamepad\_id\_ref Gamepad id

#### 4.39.3.4 pressed()

```
bool lava::gamepad::pressed (  
    gamepad_button_ref button) const [inline]
```

Check if gamepad button is pressed.

**Parameters**

<i>button</i>	Gamepad button to check
---------------	-------------------------

**Returns**

Button is pressed or not

**4.39.3.5 ready()**

```
bool lava::gamepad::ready () const
```

Check if gamepad is active.

**Returns**

Gamepad is active or not

**4.39.3.6 update()**

```
bool lava::gamepad::update ()
```

Update gamepad.

**Returns**

Update was successful or failed

**4.39.3.7 value()**

```
r32 lava::gamepad::value (  
    gamepad_axis_ref axis) const [inline]
```

Get value of axis.

**Parameters**

<i>axis</i>	Target axis
-------------	-------------

**Returns**

r32 Axis value

The documentation for this struct was generated from the following file:

- [liblava/frame/gamepad.hpp](#)

## 4.40 lava::gamepad\_manager Struct Reference

Gamepad manager.

```
#include <gamepad.hpp>
```

### Public Types

- using **listener\_func** = std::function<bool([gamepad](#), bool)>  
*Gamepad listener function.*

### Public Member Functions

- [id](#) **add** ([listener\\_func](#) listener)  
*Add listener.*
- void **remove** ([id::ref](#) func\_id)  
*Remove listener.*

### Static Public Member Functions

- static [gamepad\\_manager](#) & **singleton** ()  
*Get gamepad manager singleton.*

### 4.40.1 Detailed Description

Gamepad manager.

### 4.40.2 Member Function Documentation

#### 4.40.2.1 add()

```
id lava::gamepad_manager::add (  
    listener\_func listener)
```

Add listener.

#### Parameters

<i>listener</i>	Gamepad listener function
-----------------	---------------------------

#### Returns

id Id of function

#### 4.40.2.2 remove()

```
void lava::gamepad_manager::remove (  
    id::ref func_id)
```

Remove listener.

## Parameters

<i>func</i> ↔ <i>_id</i>	Id of function
-----------------------------	----------------

**4.40.2.3 singleton()**

```
static gamepad\_manager & lava::gamepad_manager::singleton () [inline], [static]
```

Get gamepad manager singleton.

## Returns

[gamepad\\_manager](#)& Gamepad manager

The documentation for this struct was generated from the following file:

- [liblava/frame/gamepad.hpp](#)

**4.41 lava::global\_logger Struct Reference**

Global logger.

```
#include <log.hpp>
```

**Public Member Functions**

- [s\\_logger](#) *get* ()  
*Get logger.*
- void [set](#) ([lava::s\\_logger](#) l)  
*Set logger.*
- void **reset** ()  
*Reset logger.*

**Static Public Member Functions**

- static [global\\_logger](#) & [singleton](#) ()  
*Get global logger singleton.*

**4.41.1 Detailed Description**

Global logger.

## 4.41.2 Member Function Documentation

### 4.41.2.1 get()

```
s_logger lava::global_logger::get () [inline]
```

Get logger.

#### Returns

s\_logger Logger

### 4.41.2.2 set()

```
void lava::global_logger::set (  
    lava::s_logger l) [inline]
```

Set logger.

#### Parameters

/	Logger
---	--------

### 4.41.2.3 singleton()

```
static global_logger & lava::global_logger::singleton () [inline], [static]
```

Get global logger singleton.

#### Returns

global\_logger& Global logger

The documentation for this struct was generated from the following file:

- liblava/util/[log.hpp](#)

## 4.42 lava::hex\_cell Struct Reference

Hex cell.

```
#include <hex.hpp>
```

## Public Types

- using **list** = std::vector<hex\_cell>  
*List of hex cells.*
- using **pair** = std::pair<i32, i32>  
*Hex pair (Q and R)*
- using **map** = std::unordered\_map<pair, index, pair\_hash>  
*Map of hex cells.*

## Public Member Functions

- auto **operator**<=> (hex\_cell const &) const =default  
*Compare operator.*
- **pair\_to\_pair** () const  
*Get the pair.*
- void **add** (hex\_cell const &cell)  
*Add hex cell.*
- void **subtract** (hex\_cell const &cell)  
*Subtract hex cell.*
- void **scale** (i32 factor)  
*Scale the hex cell.*
- void **rotate\_left** ()  
*Rotate to left.*
- void **rotate\_right** ()  
*Rotate to right.*

## Public Attributes

- **i32 q** {}  
*Q axis.*
- **i32 r** {}  
*R axis.*
- **i32 s** {}  
*S axis.*

### 4.42.1 Detailed Description

Hex cell.

### 4.42.2 Member Function Documentation

#### 4.42.2.1 add()

```
void lava::hex_cell::add (
    hex_cell const & cell) [inline]
```

Add hex cell.

## Parameters

<i>cell</i>	Another hex cell
-------------	------------------

**4.42.2.2 scale()**

```
void lava::hex_cell::scale (  
    i32 factor) [inline]
```

Scale the hex cell.

## Parameters

<i>factor</i>	Scaling factor
---------------	----------------

**4.42.2.3 subtract()**

```
void lava::hex_cell::subtract (  
    hex_cell const & cell) [inline]
```

Subtract hex cell.

## Parameters

<i>cell</i>	Another hex cell
-------------	------------------

**4.42.2.4 to\_pair()**

```
pair lava::hex_cell::to_pair () const [inline]
```

Get the pair.

## Returns

pair Hex pair

The documentation for this struct was generated from the following file:

- liblava/util/[hex.hpp](#)

**4.43 lava::hex\_fractional\_cell Struct Reference**

Hex fractional cell.

```
#include <hex.hpp>
```

**Public Attributes**

- [r32 q](#) {}  
*Q axis.*
- [r32 r](#) {}  
*R axis.*
- [r32 s](#) {}  
*S axis.*

**4.43.1 Detailed Description**

Hex fractional cell.

The documentation for this struct was generated from the following file:

- [liblava/util/hex.hpp](#)

**4.44 lava::hex\_grid Struct Reference**

Hex grid.

```
#include <hex.hpp>
```

**Public Member Functions**

- [hex\\_grid](#) ([r32 radius](#)=[hex\\_default\\_outer\\_radius](#))  
*Construct a new hex grid.*
- void [update](#) ([hex\\_orientation](#) orientation=[hex\\_layout\\_point\\_y](#))  
*Update the hex grid.*
- [hex\\_cell find](#) ([r32 x](#), [r32 y](#)) const  
*Find the hex cell from X and Y coordinates.*
- [hex\\_point to\\_pixel](#) ([hex\\_cell](#) const &cell) const  
*Get the hex point from hex cell.*

**Public Attributes**

- [r32 inner\\_radius](#) = 0.f  
*Hex inner radius.*
- [r32 outer\\_radius](#) = [hex\\_default\\_outer\\_radius](#)  
*Hex outer radius.*
- [hex\\_layout layout](#)  
*Hex layout.*

**4.44.1 Detailed Description**

Hex grid.

**4.44.2 Constructor & Destructor Documentation****4.44.2.1 hex\_grid()**

```
lava::hex_grid::hex_grid (
    r32 radius = hex\_default\_outer\_radius) [inline]
```

Construct a new hex grid.



## Parameters

<i>radius</i>	Hex outer radius
---------------	------------------

### 4.44.3 Member Function Documentation

#### 4.44.3.1 find()

```
hex_cell lava::hex_grid::find (  
    r32 x,  
    r32 y) const [inline]
```

Find the hex cell from X and Y coordinates.

## Parameters

<i>x</i>	X coordinate
<i>y</i>	Y coordinate

## Returns

[hex\\_cell](#) Hex cell

#### 4.44.3.2 to\_pixel()

```
hex_point lava::hex_grid::to_pixel (  
    hex_cell const & cell) const [inline]
```

Get the hex point from hex cell.

## Parameters

<i>cell</i>	Hex cell
-------------	----------

## Returns

[hex\\_point](#) Hex point

#### 4.44.3.3 update()

```
void lava::hex_grid::update (  
    hex_orientation orientation = hex_layout_point_y) [inline]
```

Update the hex grid.

## Parameters

<i>orientation</i>	Hex orientation
--------------------	-----------------

The documentation for this struct was generated from the following file:

- [liblava/util/hex.hpp](#)

## 4.45 `lava::hex_layout` Struct Reference

Hex layout.

```
#include <hex.hpp>
```

## Public Attributes

- [hex\\_orientation](#) **orientation**  
*Hex orientation.*
- [hex\\_point](#) **origin**  
*Hex origin.*
- [hex\\_point](#) **size**  
*Hex size.*

### 4.45.1 Detailed Description

Hex layout.

The documentation for this struct was generated from the following file:

- [liblava/util/hex.hpp](#)

## 4.46 `lava::hex_offset_coord` Struct Reference

Hex offset coordinates.

```
#include <hex.hpp>
```

## Public Attributes

- [i32](#) **col** {}  
*Column coordinate.*
- [i32](#) **row** {}  
*Row coordinate.*

### 4.46.1 Detailed Description

Hex offset coordinates.

The documentation for this struct was generated from the following file:

- [liblava/util/hex.hpp](#)

## 4.47 lava::hex\_orientation Struct Reference

Hex orientation.

```
#include <hex.hpp>
```

### Public Attributes

- [r32 f0](#) {}  
*F0 value.*
- [r32 f1](#) {}  
*F1 value.*
- [r32 f2](#) {}  
*F2 value.*
- [r32 f3](#) {}  
*F3 value.*
- [r32 b0](#) {}  
*B0 value.*
- [r32 b1](#) {}  
*B1 value.*
- [r32 b2](#) {}  
*B2 value.*
- [r32 b3](#) {}  
*B3 value.*
- [r32 start\\_angle](#) {}  
*Start angle.*

### 4.47.1 Detailed Description

Hex orientation.

The documentation for this struct was generated from the following file:

- [liblava/util/hex.hpp](#)

## 4.48 lava::hex\_point Struct Reference

Hex point.

```
#include <hex.hpp>
```

## Public Types

- using **list** = std::vector<[hex\\_point](#)>  
*List of hex points.*

## Public Attributes

- [r32](#) **x** {}  
*X coordinate.*
- [r32](#) **y** {}  
*Y coordinate.*

### 4.48.1 Detailed Description

Hex point.

The documentation for this struct was generated from the following file:

- liblava/util/[hex.hpp](#)

## 4.49 lava::imgui::icon\_font Struct Reference

ImGui icon font settings.

```
#include <imgui.hpp>
```

## Public Attributes

- [data](#) **font\_data**  
*Icon font data.*
- [ui16](#) **range\_begin** = 0  
*Icon range begin.*
- [ui16](#) **range\_end** = 0  
*Icon range end.*
- [r32](#) **size** = [default\\_imgui\\_font\\_size](#)  
*Default icon font size.*

### 4.49.1 Detailed Description

ImGui icon font settings.

The documentation for this struct was generated from the following file:

- liblava/app/[imgui.hpp](#)

## 4.50 `lava::id` Struct Reference

Identification.

```
#include <id.hpp>
```

### Public Types

- using **ref** = `id` const&  
*Reference to id.*
- using **set** = `std::set<id>`  
*Set of ids.*
- using **set\_ref** = `set` const&  
*Reference to set of ids.*
- using **list** = `std::vector<id>`  
*List of ids.*
- using **map** = `std::map<id, id>`  
*Map of ids.*
- using **index\_map** = `std::map<id, index>`  
*Index map by ids.*
- using **string\_map** = `std::map<id, string>`  
*String map by ids.*

### Public Member Functions

- **id** ()=default  
*Construct a new id.*
- **id** (index value)  
*Construct a new id.*
- bool **valid** () const  
*Check if the id is valid.*
- **string to\_string** () const  
*Convert the id to string.*
- void **invalidate** ()  
*Invalidate id.*
- auto **operator<=>** (id const &) const =default  
*Compare operator.*

### Public Attributes

- **index value** = `no_index`  
*Value.*

### 4.50.1 Detailed Description

Identification.

### 4.50.2 Constructor & Destructor Documentation

#### 4.50.2.1 `id()`

```
lava::id::id (  
    index value) [inline]
```

Construct a new id.

## Parameters

<i>value</i>	Value of id
--------------	-------------

### 4.50.3 Member Function Documentation

#### 4.50.3.1 to\_string()

```
string lava::id::to_string () const [inline]
```

Convert the id to string.

## Returns

string String representation of id

#### 4.50.3.2 valid()

```
bool lava::id::valid () const [inline]
```

Check if the id is valid.

## Returns

Id is valid or not

The documentation for this struct was generated from the following file:

- liblava/core/[id.hpp](#)

## 4.51 lava::id\_listeners< T > Struct Template Reference

Id listeners.

```
#include <id.hpp>
```

### Public Member Functions

- [id add](#) (typename T::func const &listener)  
*Add listener to map.*
- void [remove](#) ([id](#) &[id](#))  
*Remove listener from map by id.*
- T::listeners const & [get\\_list](#) () const  
*Get the list.*

#### 4.51.1 Detailed Description

```
template<typename T>
struct lava::id_listeners< T >
```

Id listeners.

## Template Parameters

<i>T</i>	Listener
----------	----------

## 4.51.2 Member Function Documentation

### 4.51.2.1 `add()`

```
template<typename T >
id lava::id_listeners< T >::add (
    typename T::func const & listener) [inline]
```

Add listener to map.

## Parameters

<i>listener</i>	Target listener
-----------------	-----------------

## Returns

id Id of listener

### 4.51.2.2 `get_list()`

```
template<typename T >
T::listeners const & lava::id_listeners< T >::get_list () const [inline]
```

Get the list.

## Returns

T::listeners const& List of listeners

### 4.51.2.3 `remove()`

```
template<typename T >
void lava::id_listeners< T >::remove (
    id & id) [inline]
```

Remove listener from map by id.

## Parameters

<i>id</i>	Id of listener
-----------	----------------

The documentation for this struct was generated from the following file:

- [liblava/core/id.hpp](#)

## 4.52 `lava::id_registry< T, Meta >` Struct Template Reference

Id registry.

```
#include <id.hpp>
```

### Public Types

- using **s\_ptr** = std::shared\_ptr<T>  
*Shared pointer to id registry.*
- using **s\_map** = std::map<id, s\_ptr>  
*Map of id registries.*
- using **meta\_map** = std::map<id, Meta>  
*Map of ids with meta.*

### Public Member Functions

- **id create** (Meta info={})  
*Create a new object in registry.*
- void **add** (s\_ptr object, Meta info={})  
*Add a object with meta to registry.*
- bool **exists** (id::ref object\_id) const  
*Check if object exists in registry.*
- s\_ptr **get** (id::ref object\_id) const  
*Get the object by id.*
- Meta const & **get\_meta** (id::ref object\_id) const  
*Get the meta by id.*
- s\_map const & **get\_all** () const  
*Get all objects.*
- meta\_map const & **get\_all\_meta** () const  
*Get all meta objects.*
- bool **update** (id::ref object\_id, Meta const &meta)  
*Update meta of object.*
- void **remove** (id::ref object\_id)  
*Remove object from registry.*
- void **clear** ()  
*Clear the registry.*

### 4.52.1 Detailed Description

```
template<typename T, typename Meta>
struct lava::id_registry< T, Meta >
```

Id registry.

#### Template Parameters

<i>T</i>	Type of objects hold in registry
<i>Meta</i>	Meta type for object



## 4.52.2 Member Function Documentation

### 4.52.2.1 add()

```
template<typename T , typename Meta >
void lava::id_registry< T, Meta >::add (
    s_ptr object,
    Meta info = {}) [inline]
```

Add a object with meta to registry.

#### Parameters

<i>object</i>	Object to add
<i>info</i>	Meta of object

### 4.52.2.2 create()

```
template<typename T , typename Meta >
id lava::id_registry< T, Meta >::create (
    Meta info = {}) [inline]
```

Create a new object in registry.

#### Parameters

<i>info</i>	Meta information
-------------	------------------

#### Returns

id Object id

### 4.52.2.3 exists()

```
template<typename T , typename Meta >
bool lava::id_registry< T, Meta >::exists (
    id::ref object_id) const [inline]
```

Check if object exists in registry.

#### Parameters

<i>object↔ _id</i>	Object to check
------------------------	-----------------

#### Returns

Object exists or not

### 4.52.2.4 get()

```
template<typename T , typename Meta >
s_ptr lava::id_registry< T, Meta >::get (
    id::ref object_id) const [inline]
```

Get the object by id.

**Parameters**

<i>object</i> ↔ _id	Object id
------------------------	-----------

**Returns**

s\_ptr Shared pointer to object

**4.52.2.5 get\_all()**

```
template<typename T , typename Meta >
s_map const & lava::id_registry< T, Meta >::get_all () const [inline]
```

Get all objects.

**Returns**

s\_map const& Map with objects

**4.52.2.6 get\_all\_meta()**

```
template<typename T , typename Meta >
meta_map const & lava::id_registry< T, Meta >::get_all_meta () const [inline]
```

Get all meta objects.

**Returns**

meta\_map const& Map with metas

**4.52.2.7 get\_meta()**

```
template<typename T , typename Meta >
Meta const & lava::id_registry< T, Meta >::get_meta (
    id::ref object_id) const [inline]
```

Get the meta by id.

**Parameters**

<i>object</i> ↔ _id	Object id
------------------------	-----------

**Returns**

Meta Meta object

**4.52.2.8 remove()**

```
template<typename T , typename Meta >
void lava::id_registry< T, Meta >::remove (
    id::ref object_id) [inline]
```

Remove object from registry.

## Parameters

<i>object</i> ↔ <i>_id</i>	Object id
-------------------------------	-----------

## 4.52.2.9 update()

```
template<typename T , typename Meta >
bool lava::id_registry< T, Meta >::update (
    id::ref object_id,
    Meta const & meta) [inline]
```

Update meta of object.

## Parameters

<i>object</i> ↔ <i>_id</i>	Object id
<i>meta</i>	Meta to update

## Returns

Meta updated or not

The documentation for this struct was generated from the following file:

- [liblava/core/id.hpp](#)

## 4.53 lava::ids Struct Reference

Id factory.

```
#include <id.hpp>
```

## Public Member Functions

- [id next](#) ()  
*Get next id from factory.*

## Static Public Member Functions

- static [ids](#) & [instance](#) ()  
*Get id factory instance.*

### 4.53.1 Detailed Description

Id factory.

### 4.53.2 Member Function Documentation

#### 4.53.2.1 instance()

```
static ids & lava::ids::instance () [inline], [static]
```

Get id factory instance.

Returns

ids& Id factory

#### 4.53.2.2 next()

```
id lava::ids::next () [inline]
```

Get next id from factory.

Returns

id Next id

The documentation for this struct was generated from the following file:

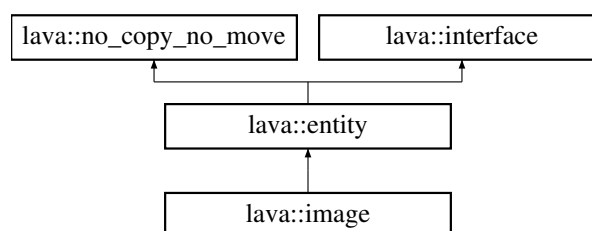
- [liblava/core/id.hpp](#)

## 4.54 lava::image Struct Reference

Image.

```
#include <image.hpp>
```

Inheritance diagram for lava::image:



## Public Types

- using **s\_ptr** = std::shared\_ptr<image>  
*Shared pointer to image.*
- using **s\_map** = std::map<id, s\_ptr>  
*Map of images.*
- using **s\_list** = std::vector<s\_ptr>  
*List of images.*

## Public Member Functions

- **image** (VkFormat format, VkImage vk\_image=0)  
*Construct a new image.*
- bool **create** (device::ptr device, uv2 size, VmaMemoryUsage memory\_usage=VMA\_MEMORY\_USAGE\_GPU\_ONLY, VmaAllocationCreateFlags allocation\_flags=0)  
*Create a new image.*
- void **destroy** (bool view\_only=false)  
*Destroy the image.*
- void **destroy\_view** ()  
*Destroy the image view.*
- device::ptr **get\_device** ()  
*Get the device.*
- VkFormat **get\_format** () const  
*Get the format of the image.*
- uv2 **get\_size** () const  
*Get the size of the image.*
- ui32 **get\_depth** () const  
*Get the depth of the image.*
- VkImage **get** () const  
*Get the image.*
- VkImageView **get\_view** () const  
*Get the image view.*
- VkImageCreateInfo const & **get\_info** () const  
*Get the image create information.*
- VkImageViewCreateInfo const & **get\_view\_info** () const  
*Get the image view create information.*
- VkImageSubresourceRange const & **get\_subresource\_range** () const  
*Get the subresource range of the image.*
- void **set\_flags** (VkImageCreateFlagBits flags)  
*Set the image create flags.*
- void **set\_tiling** (VkImageTiling tiling)  
*Set the image tiling.*
- void **set\_usage** (VkImageUsageFlags usage)  
*Set the image usage.*
- void **set\_layout** (VkImageLayout initial)  
*Set the initial layout of the image.*
- void **set\_aspect\_mask** (VkImageAspectFlags aspectMask)  
*Set the aspect mask of the image.*
- void **set\_level\_count** (ui32 levels)  
*Set the level count of the image.*

- void [set\\_layer\\_count](#) (ui32 layers)  
*Set the layer count of the image.*
- void [set\\_component](#) (VkComponentMapping mapping={})  
*Set the component mapping of the image.*
- void [set\\_view\\_type](#) (VkImageViewType type)  
*Set the view type of the image.*
- VmaAllocation const & [get\\_allocation](#) () const  
*Get the allocation of the image.*

### Public Member Functions inherited from [lava::entity](#)

- [entity](#) ()  
*Construct a new entity.*
- [id::ref get\\_id](#) () const  
*Get the id of entity.*

### Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

### Public Member Functions inherited from [lava::interface](#)

- virtual [~interface](#) ()=default  
*Destroy the interface.*

### Static Public Member Functions

- static [s\\_ptr make](#) (VkFormat format, VkImage vk\_image=0)  
*Make a new image.*

## 4.54.1 Detailed Description

Image.

## 4.54.2 Constructor & Destructor Documentation

### 4.54.2.1 [image\(\)](#)

```
lava::image::image (
    VkFormat format,
    VkImage vk_image = 0) [explicit]
```

Construct a new image.

## Parameters

<i>format</i>	Image format
<i>vk_image</i>	Vulkan image

### 4.54.3 Member Function Documentation

#### 4.54.3.1 create()

```
bool lava::image::create (
    device::ptr device,
    uv2 size,
    VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_GPU_ONLY,
    VmaAllocationCreateFlags allocation_flags = 0)
```

Create a new image.

## Parameters

<i>device</i>	Vulkan device
<i>size</i>	Image size
<i>memory_usage</i>	Memory usage
<i>allocation_flags</i>	Allocation flags

## Returns

Create was successful or failed

#### 4.54.3.2 destroy()

```
void lava::image::destroy (
    bool view_only = false)
```

Destroy the image.

## Parameters

<i>view_only</i>	Destroy only the image view
------------------	-----------------------------

#### 4.54.3.3 get()

```
VkImage lava::image::get () const [inline]
```

Get the image.

## Returns

VkImage Vulkan image

#### 4.54.3.4 get\_allocation()

```
VmaAllocation const & lava::image::get_allocation () const [inline]
```

Get the allocation of the image.

##### Returns

VmaAllocation const& Image allocation

#### 4.54.3.5 get\_depth()

```
ui32 lava::image::get_depth () const [inline]
```

Get the depth of the image.

##### Returns

ui32 Image depth

#### 4.54.3.6 get\_device()

```
device::ptr lava::image::get_device () [inline]
```

Get the device.

##### Returns

device::ptr Vulkan device

#### 4.54.3.7 get\_format()

```
VkFormat lava::image::get_format () const [inline]
```

Get the format of the image.

##### Returns

VkFormat Image format

#### 4.54.3.8 get\_info()

```
VkImageCreateInfo const & lava::image::get_info () const [inline]
```

Get the image create information.

##### Returns

VkImageCreateInfo const& Image create information



#### 4.54.3.9 get\_size()

```
uv2 lava::image::get_size () const [inline]
```

Get the size of the image.

##### Returns

uv2 Image size

#### 4.54.3.10 get\_subresource\_range()

```
VkImageSubresourceRange const & lava::image::get_subresource_range () const [inline]
```

Get the subresource range of the image.

##### Returns

VkImageSubresourceRange const& Image subresource range

#### 4.54.3.11 get\_view()

```
VkImageView lava::image::get_view () const [inline]
```

Get the image view.

##### Returns

VkImageView Vulkan image view

#### 4.54.3.12 get\_view\_info()

```
VkImageViewCreateInfo const & lava::image::get_view_info () const [inline]
```

Get the image view create information.

##### Returns

VkImageViewCreateInfo const& Image view create information

#### 4.54.3.13 make()

```
static s_ptr lava::image::make (  
    VkFormat format,  
    VkImage vk_image = 0) [inline], [static]
```

Make a new image.

## Parameters

<i>format</i>	Image format
<i>vk_image</i>	Vulkan image

## Returns

s\_ptr Shared pointer to image

**4.54.3.14 set\_aspect\_mask()**

```
void lava::image::set_aspect_mask (
    VkImageAspectFlags aspectMask) [inline]
```

Set the aspect mask of the image.

## Parameters

<i>aspectMask</i>	Image aspect flags
-------------------	--------------------

**4.54.3.15 set\_component()**

```
void lava::image::set_component (
    VkComponentMapping mapping = {}) [inline]
```

Set the component mapping of the image.

## Parameters

<i>mapping</i>	Component mapping
----------------	-------------------

**4.54.3.16 set\_flags()**

```
void lava::image::set_flags (
    VkImageCreateFlagBits flags) [inline]
```

Set the image create flags.

## Parameters

<i>flags</i>	Image create flag bits
--------------	------------------------

**4.54.3.17 set\_layer\_count()**

```
void lava::image::set_layer_count (
    ui32 layers) [inline]
```

Set the layer count of the image.

## Parameters

<i>layers</i>	Number of layers
---------------	------------------

**4.54.3.18 set\_layout()**

```
void lava::image::set_layout (
    VkImageLayout initial) [inline]
```

Set the initial layout of the image.

## Parameters

<i>initial</i>	Initial image layout
----------------	----------------------

**4.54.3.19 set\_level\_count()**

```
void lava::image::set_level_count (
    ui32 levels) [inline]
```

Set the level count of the image.

## Parameters

<i>levels</i>	Number of levels
---------------	------------------

**4.54.3.20 set\_tiling()**

```
void lava::image::set_tiling (
    VkImageTiling tiling) [inline]
```

Set the image tiling.

## Parameters

<i>tiling</i>	Image tiling
---------------	--------------

**4.54.3.21 set\_usage()**

```
void lava::image::set_usage (
    VkImageUsageFlags usage) [inline]
```

Set the image usage.

## Parameters

<i>usage</i>	Image usage flags
--------------	-------------------

**4.54.3.22 set\_view\_type()**

```
void lava::image::set_view_type (
    VkImageViewType type) [inline]
```

Set the view type of the image.

## Parameters

<i>type</i>	Image view type
-------------	-----------------

The documentation for this struct was generated from the following file:

- [liblava/resource/image.hpp](#)

**4.55 lava::image\_data Struct Reference**

Image data.

```
#include <image.hpp>
```

**Public Types**

- using **s\_ptr** = std::shared\_ptr<[image\\_data](#)>  
*Shared pointer to image data.*

**Public Member Functions**

- bool [ready](#) () const  
*Check if image data is ready.*
- [data::ptr get\\_data](#) ()  
*Get image data.*
- void [set\\_data](#) (data::ptr data)  
*Set image data.*
- [size\\_t size](#) () const  
*Get image data size.*
- [~image\\_data](#) ()  
*Destroy the image data.*

## Public Attributes

- **uv2 dimensions** = `uv2(0, 0)`  
*Dimensions.*
- **ui32 channels** = 0  
*Number of channels.*

### 4.55.1 Detailed Description

Image data.

### 4.55.2 Member Function Documentation

#### 4.55.2.1 get\_data()

```
data::ptr lava::image_data::get_data () [inline]
```

Get image data.

#### Returns

`data::ptr` Pointer to image data

#### 4.55.2.2 ready()

```
bool lava::image_data::ready () const [inline]
```

Check if image data is ready.

#### Returns

Image data is ready or not

#### 4.55.2.3 set\_data()

```
void lava::image_data::set_data (  
    data::ptr data) [inline]
```

Set image data.

#### Parameters

<i>data</i>	Pointer to image data
-------------	-----------------------

#### 4.55.2.4 size()

```
size_t lava::image_data::size () const [inline]
```

Get image data size.

##### Returns

size\_t Image data size

The documentation for this struct was generated from the following file:

- [liblava/resource/image.hpp](#)

## 4.56 lava::imgui Struct Reference

ImGui integration.

```
#include <imgui.hpp>
```

### Classes

- struct [config](#)  
*ImGui configuration.*
- struct [font](#)  
*ImGui font settings.*
- struct [icon\\_font](#)  
*ImGui icon font settings.*

### Public Types

- using **ptr** = [imgui\\*](#)  
*Pointer to imgui.*
- using **draw\_func** = std::function<void()>  
*Draw function.*

## Public Member Functions

- **imgui** ()=default  
*Construct a new ImGui.*
- **imgui** (GLFWwindow \***window**)  
*Construct a new ImGui.*
- **~imgui** ()  
*Destroy the ImGui.*
- void **setup** (GLFWwindow \***window**, **config** **config**)  
*Set up ImGui with configuration.*
- void **setup** (GLFWwindow \***win**)  
*Set up default ImGui.*
- bool **create** (**render\_pipeline::s\_ptr** **pipeline**, **index** **max\_frames**)  
*Create pipeline for ImGui.*
- bool **create** (**device::ptr** **dev**, **index** **frames**, VkPipelineCache **pipeline\_cache**)  
*Create pipeline for ImGui with device.*
- bool **create** (**device::ptr** **dev**, **index** **frames**, VkRenderPass **pass**, VkPipelineCache **pipeline\_cache**=0)  
*Create pipeline for ImGui with device and render pass.*
- bool **upload\_fonts** (**texture::s\_ptr** **texture**)  
*Upload font texture.*
- void **destroy** ()  
*Destroy ImGui.*
- bool **ready** () const  
*Check if ImGui is ready.*
- **render\_pipeline::s\_ptr** **get\_pipeline** ()  
*Get the pipeline.*
- bool **capture\_mouse** () const  
*Check if mouse capture is active.*
- bool **capture\_keyboard** () const  
*Check if keyboard capture is active.*
- void **set\_active** (bool **value**=true)  
*Set ImGui active.*
- bool **activated** () const  
*Check if ImGui is activated.*
- void **toggle** ()  
*Togge active state.*
- void **set\_ini\_file** (std::filesystem::path **dir**)  
*Set the ini file.*
- std::filesystem::path **get\_ini\_file** () const  
*Get the ini file.*
- void **convert\_style\_to\_srgb** ()  
*Convert style to sRGB.*
- **input\_callback** const & **get\_input\_callback** () const  
*Get the input callback.*

## Public Attributes

- **draw\_func** **on\_draw**  
*Function called on ImGui draw.*
- **layer\_list** **layers**  
*Layer list.*

### 4.56.1 Detailed Description

ImGui integration.

### 4.56.2 Constructor & Destructor Documentation

#### 4.56.2.1 `imgui()`

```
lava::imgui::imgui (
    GLFWwindow * window) [inline], [explicit]
```

Construct a new ImGui.

##### Parameters

<i>window</i>	Window for ImGui
---------------	------------------

### 4.56.3 Member Function Documentation

#### 4.56.3.1 `activated()`

```
bool lava::imgui::activated () const [inline]
```

Check if ImGui is activated.

##### Returns

ImGui is active or not

#### 4.56.3.2 `capture_keyboard()`

```
bool lava::imgui::capture_keyboard () const
```

Check if keyboard capture is active.

##### Returns

Capture is active or not

#### 4.56.3.3 `capture_mouse()`

```
bool lava::imgui::capture_mouse () const
```

Check if mouse capture is active.

##### Returns

Capture is active or not

#### 4.56.3.4 `create()` [1/3]

```
bool lava::imgui::create (
    device::ptr dev,
    index frames,
    VkPipelineCache pipeline_cache) [inline]
```

Create pipeline for ImGui with device.



## Parameters

<i>dev</i>	Vulkan device
<i>frames</i>	Number of frames
<i>pipeline_cache</i>	Pipeline cache

## Returns

Create was successful or failed

**4.56.3.5 create() [2/3]**

```
bool lava::imgui::create (  
    device::ptr dev,  
    index frames,  
    VkRenderPass pass,  
    VkPipelineCache pipeline_cache = 0) [inline]
```

Create pipeline for ImGui with device and render pass.

## Parameters

<i>dev</i>	Vulkan device
<i>frames</i>	Number of frames
<i>pass</i>	Render pass
<i>pipeline_cache</i>	Pipeline cache

## Returns

Create was successful or failed

**4.56.3.6 create() [3/3]**

```
bool lava::imgui::create (  
    render_pipeline::s_ptr pipeline,  
    index max_frames)
```

Create pipeline for ImGui.

## Parameters

<i>pipeline</i>	Render pipeline
<i>max_frames</i>	Number of frames

## Returns

Create was successful or failed

#### 4.56.3.7 get\_ini\_file()

```
std::filesystem::path lava::imgui::get_ini_file () const [inline]
```

Get the ini file.

Returns

fs::path Path of file

#### 4.56.3.8 get\_input\_callback()

```
input_callback const & lava::imgui::get_input_callback () const [inline]
```

Get the input callback.

Returns

input\_callback const& Input callback

#### 4.56.3.9 get\_pipeline()

```
render_pipeline::s_ptr lava::imgui::get_pipeline () [inline]
```

Get the pipeline.

Returns

render\_pipeline::s\_ptr Render pipeline

#### 4.56.3.10 ready()

```
bool lava::imgui::ready () const [inline]
```

Check if ImGui is ready.

Returns

ImGui is ready or not

#### 4.56.3.11 set\_active()

```
void lava::imgui::set_active (  
    bool value = true) [inline]
```

Set ImGui active.

## Parameters

<i>value</i>	Active state
--------------	--------------

**4.56.3.12 set\_ini\_file()**

```
void lava::imgui::set_ini_file (  
    std::filesystem::path dir)
```

Set the ini file.

## Parameters

<i>dir</i>	Path for file
------------	---------------

**4.56.3.13 setup() [1/2]**

```
void lava::imgui::setup (  
    GLFWwindow * win) [inline]
```

Set up default ImGui.

## Parameters

<i>win</i>	Target window
------------	---------------

**4.56.3.14 setup() [2/2]**

```
void lava::imgui::setup (  
    GLFWwindow * window,  
    config config)
```

Set up ImGui with configuration.

## Parameters

<i>window</i>	Target window
<i>config</i>	Configuration

**4.56.3.15 upload\_fonts()**

```
bool lava::imgui::upload_fonts (  
    texture::s_ptr texture)
```

Upload font texture.

## Parameters

<i>texture</i>	Texture to upload
----------------	-------------------

## Returns

Upload was successful or failed

The documentation for this struct was generated from the following file:

- [liblava/app/imgui.hpp](#)

## 4.57 lava::input Struct Reference

Input handling.

```
#include <input.hpp>
```

## Public Types

- using **ptr** = [input\\*](#)  
*Pointer to input.*

## Public Member Functions

- void **handle\_events** ()  
*Handle events.*
- void **add** ([input\\_callback::cptr](#) callback)  
*Add callback to the input handling.*
- void **remove** ([input\\_callback::cptr](#) callback)  
*Remove callback from the input handling.*
- [mouse\\_position\\_ref](#) **get\_mouse\_position** () const  
*Get the mouse position.*
- void **set\_mouse\_position** ([mouse\\_position\\_ref](#) position)  
*Set the mouse position.*

## Public Attributes

- [input\\_key\\_events](#) **key**  
*List of key events.*
- [input\\_scroll\\_events](#) **scroll**  
*List of scroll events.*
- [input\\_mouse\\_move\\_events](#) **mouse\_move**  
*List of mouse move events.*
- [input\\_mouse\\_button\\_events](#) **mouse\_button**  
*List of mouse button events.*
- [input\\_mouse\\_active\\_events](#) **mouse\_active**  
*List of mouse active events.*
- [input\\_path\\_drop\\_events](#) **path\_drop**  
*List of path drop events.*

### 4.57.1 Detailed Description

Input handling.

### 4.57.2 Member Function Documentation

#### 4.57.2.1 add()

```
void lava::input::add (
    input_callback::cptr callback) [inline]
```

Add callback to the input handling.

##### Parameters

<i>callback</i>	Callback to add
-----------------	-----------------

#### 4.57.2.2 get\_mouse\_position()

```
mouse_position_ref lava::input::get_mouse_position () const [inline]
```

Get the mouse position.

##### Returns

mouse\_position\_ref Current mouse position

#### 4.57.2.3 remove()

```
void lava::input::remove (
    input_callback::cptr callback) [inline]
```

Remove callback from the input handling.

##### Parameters

<i>callback</i>	Callback to remove
-----------------	--------------------

#### 4.57.2.4 set\_mouse\_position()

```
void lava::input::set_mouse_position (
    mouse_position_ref position) [inline]
```

Set the mouse position.

## Parameters

<i>position</i>	Current mouse position
-----------------	------------------------

The documentation for this struct was generated from the following file:

- [liblava/frame/input.hpp](#)

## 4.58 lava::input\_callback Struct Reference

Input callback.

```
#include <input.hpp>
```

### Public Types

- using **cptr** = [input\\_callback](#) const\*  
*Const pointer to input callback.*
- using **list** = std::vector<[input\\_callback\\*](#)>  
*List of input callbacks.*
- using **clist** = std::vector<[cptr](#)>  
*List of const input callbacks.*
- template<typename T >  
using **func** = std::function<bool(typename T::ref)>  
*Input callback functions.*

### Public Attributes

- [key\\_event::func](#) **on\_key\_event**  
*Called on key event.*
- [scroll\\_event::func](#) **on\_scroll\_event**  
*Called on scroll event.*
- [mouse\\_move\\_event::func](#) **on\_mouse\_move\_event**  
*Called on mouse move event.*
- [mouse\\_button\\_event::func](#) **on\_mouse\_button\_event**  
*Called on mouse button event.*
- [mouse\\_active\\_event::func](#) **on\_mouse\_active\_event**  
*Called on mouse active event.*
- [path\\_drop\\_event::func](#) **on\_path\_drop\_event**  
*Called on path drop event.*

### 4.58.1 Detailed Description

Input callback.

### 4.58.2 Member Typedef Documentation

#### 4.58.2.1 func

```
template<typename T >
using lava::input\_callback::func = std::function<bool(typename T::ref)>
```

Input callback functions.

## Template Parameters

<code>T</code>	Type of callback
----------------	------------------

The documentation for this struct was generated from the following file:

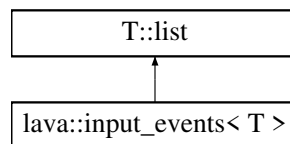
- [liblava/frame/input.hpp](#)

## 4.59 `lava::input_events< T >` Struct Template Reference

List of input events.

```
#include <input.hpp>
```

Inheritance diagram for `lava::input_events< T >`:



## Public Member Functions

- void [add](#) (typename `T::ref event`)  
*Add event to list.*

## Public Attributes

- [id\\_listeners< T > listeners](#)  
*List of event listeners.*

### 4.59.1 Detailed Description

```
template<typename T>
struct lava::input_events< T >
```

List of input events.

## Template Parameters

<code>T</code>	Type of event
----------------	---------------

### 4.59.2 Member Function Documentation

#### 4.59.2.1 `add()`

```
template<typename T >
void lava::input_events< T >::add (
    typename T::ref event) [inline]
```

Add event to list.

## Parameters

<i>event</i>	Event to add
--------------	--------------

The documentation for this struct was generated from the following file:

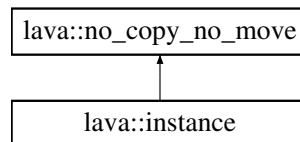
- [liblava/frame/input.hpp](#)

## 4.60 lava::instance Struct Reference

Vulkan instance.

```
#include <instance.hpp>
```

Inheritance diagram for lava::instance:



## Classes

- struct [create\\_param](#)  
*Instance create parameters.*
- struct [debug\\_config](#)  
*Debug configuration.*

### Public Member Functions

- bool [create](#) ([create\\_param](#) &param, [debug\\_config::ref](#) debug, [instance\\_info::ref](#) info)  
*Create a new instance.*
- void **destroy** ()  
*Destroy the instance.*
- [physical\\_device::s\\_list](#) const & [get\\_physical\\_devices](#) () const  
*Get the physical devices.*
- [physical\\_device::ref](#) [get\\_first\\_physical\\_device](#) () const  
*Get the first physical device.*
- VkInstance [get](#) () const  
*Get the Vulkan instance.*
- [debug\\_config::ref](#) [get\\_debug\\_config](#) () const  
*Get the debug configuration.*
- [instance\\_info::ref](#) [get\\_info](#) () const  
*Get the instance information.*



## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Static Public Member Functions

- static [instance](#) & [singleton](#) ()  
*Instance singleton.*

### 4.60.1 Detailed Description

Vulkan instance.

### 4.60.2 Member Function Documentation

#### 4.60.2.1 create()

```
bool lava::instance::create (
    create\_param & param,
    debug\_config::ref debug,
    instance\_info::ref info)
```

Create a new instance.

#### Parameters

<i>param</i>	Create parameters
<i>debug</i>	Debug configuration
<i>info</i>	Instance information

#### Returns

Create was successful or failed

#### 4.60.2.2 get()

```
VkInstance lava::instance::get () const [inline]
```

Get the Vulkan instance.

#### Returns

VkInstance Vulkan instance

#### 4.60.2.3 get\_debug\_config()

```
debug_config::ref lava::instance::get_debug_config () const [inline]
```

Get the debug configuration.

Returns

[debug\\_config::ref](#) Debug configuration

#### 4.60.2.4 get\_first\_physical\_device()

```
physical_device::ref lava::instance::get_first_physical_device () const [inline]
```

Get the first physical device.

Returns

[physical\\_device::ref](#) Physical device

#### 4.60.2.5 get\_info()

```
instance_info::ref lava::instance::get_info () const [inline]
```

Get the instance information.

Returns

[instance\\_info::ref](#) Instance information

#### 4.60.2.6 get\_physical\_devices()

```
physical_device::s_list const & lava::instance::get_physical_devices () const [inline]
```

Get the physical devices.

Returns

[physical\\_device::s\\_list](#) const& List of physical devices

#### 4.60.2.7 singleton()

```
static instance & lava::instance::singleton () [inline], [static]
```

Instance singleton.

Returns

instance& Instance

The documentation for this struct was generated from the following file:

- [liblava/base/instance.hpp](#)

## 4.61 `lava::instance_info` Struct Reference

Vulkan instance information.

```
#include <instance.hpp>
```

### Public Types

- using **ref** = `instance_info` const&  
*Reference to a instance.*

### Public Attributes

- **name** `app_name` = `_lava_`  
*Name of application.*
- **name** `engine_name` = `_liblava_`  
*Name of engine.*
- **sem\_version** `app_version`  
*Version of application.*
- **sem\_version** `engine_version`  
*Version of engine.*
- **api\_version** `req_api_version` = `api_version::v1_0`  
*Required Vulkan API version.*

### 4.61.1 Detailed Description

Vulkan instance information.

The documentation for this struct was generated from the following file:

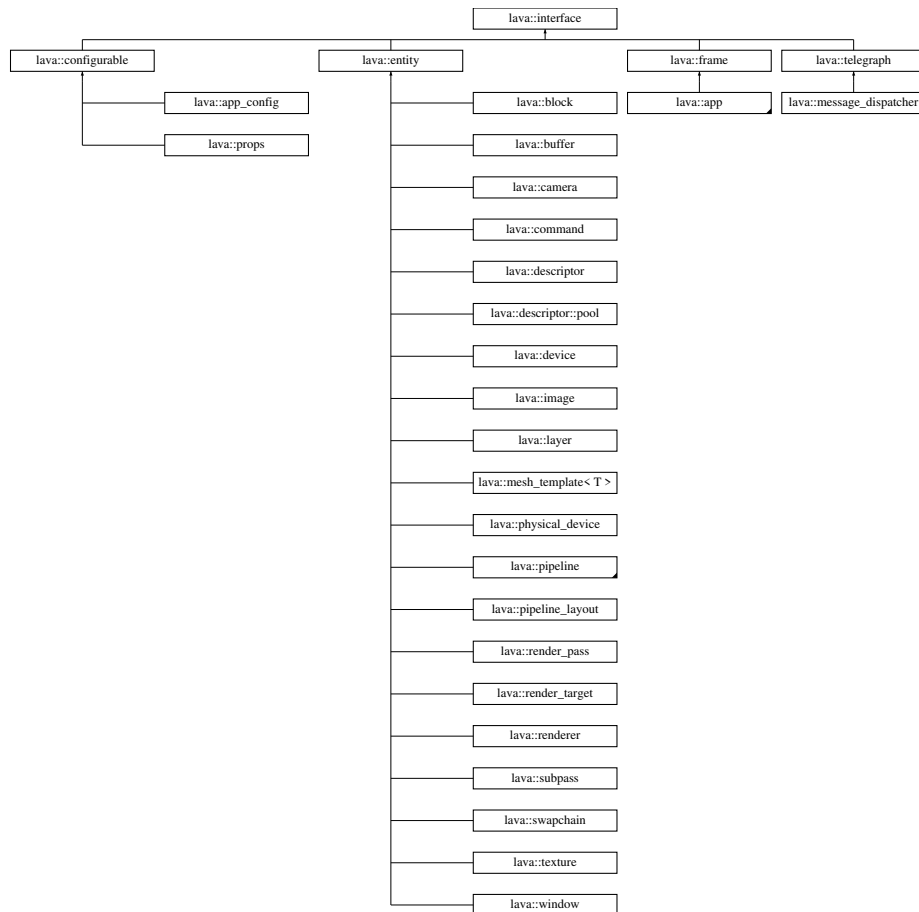
- `liblava/base/instance.hpp`

## 4.62 `lava::interface` Struct Reference

Interface.

```
#include <types.hpp>
```

Inheritance diagram for `lava::interface`:



## Public Member Functions

- virtual **~interface** ()=default  
*Destroy the interface.*

### 4.62.1 Detailed Description

Interface.

The documentation for this struct was generated from the following file:

- [liblava/core/types.hpp](#)

## 4.63 lava::props::item Struct Reference

Prop item.

```
#include <props.hpp>
```

## Public Types

- using **map** = std::map<string, item>  
*Map of items.*

## Public Member Functions

- **item** (string\_ref filename)  
*Construct a new prop.*

## Public Attributes

- **string filename**  
*File name of prop.*
- **file\_data data**  
*File data of prop.*

### 4.63.1 Detailed Description

Prop item.

### 4.63.2 Constructor & Destructor Documentation

#### 4.63.2.1 item()

```
lava::props::item::item (
    string_ref filename) [inline]
```

Construct a new prop.

#### Parameters

<i>filename</i>	File name of prop
-----------------	-------------------

The documentation for this struct was generated from the following file:

- liblava/engine/props.hpp

## 4.64 lava::json\_file Struct Reference

Json file.

```
#include <json_file.hpp>
```

## Classes

- struct [callback](#)  
*Json file callback.*

## Public Member Functions

- [json\\_file](#) ([string\\_ref](#) path="config.json")  
*Construct a new json file.*
- void [add](#) ([callback](#) \*[callback](#))  
*Add callback to json file.*
- void [remove](#) ([callback](#) \*[callback](#))  
*Remove callback from json file.*
- void [clear](#) ()  
*Clear all callbacks.*
- void [set](#) ([string\\_ref](#) value)  
*Set path of the json file.*
- [string\\_ref](#) [get](#) () const  
*Get path of the json file.*
- bool [load](#) ()  
*Load the json file.*
- bool [save](#) ()  
*Save the json file.*

### 4.64.1 Detailed Description

Json file.

### 4.64.2 Constructor & Destructor Documentation

#### 4.64.2.1 [json\\_file\(\)](#)

```
lava::json_file::json_file (
    string\_ref path = "config.json") [explicit]
```

Construct a new json file.

#### Parameters

<i>path</i>	Name of file
-------------	--------------

### 4.64.3 Member Function Documentation

#### 4.64.3.1 [add\(\)](#)

```
void lava::json_file::add (
    callback * callback)
```

Add callback to json file.

## Parameters

<i>callback</i>	Callback to add
-----------------	-----------------

**4.64.3.2 get()**

```
string_ref lava::json_file::get () const [inline]
```

Get path of the json file.

## Returns

name Name of file

**4.64.3.3 load()**

```
bool lava::json_file::load ()
```

Load the json file.

## Returns

Load was successful or failed

**4.64.3.4 remove()**

```
void lava::json_file::remove (  
    callback * callback)
```

Remove callback from json file.

## Parameters

<i>callback</i>	Callback to remove
-----------------	--------------------

**4.64.3.5 save()**

```
bool lava::json_file::save ()
```

Save the json file.

## Returns

Save was successful or failed

**4.64.3.6 set()**

```
void lava::json_file::set (  
    string_ref value) [inline]
```

Set path of the json file.

## Parameters

<i>value</i>	Name of file
--------------	--------------

The documentation for this struct was generated from the following file:

- [liblava/file/json\\_file.hpp](#)

## 4.65 lava::key\_event Struct Reference

Key event.

```
#include <input.hpp>
```

### Public Types

- using **ref** = [key\\_event](#) const&  
*Reference to key event.*
- using **func** = std::function<bool([ref](#))>  
*Key event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of key event listeners.*
- using **list** = std::vector<[key\\_event](#)>  
*List of key events.*

### Public Member Functions

- bool [pressed](#) ([key\\_ref](#) k) const  
*Check if key is pressed.*
- bool [released](#) ([key\\_ref](#) k) const  
*Check if key is released.*
- bool [repeated](#) ([key\\_ref](#) k) const  
*Check if key is repeated.*
- bool [active](#) () const  
*Check if key is active.*
- bool [pressed](#) ([key\\_ref](#) k, [mod\\_ref](#) m) const  
*Check if key is pressed with mod.*

### Public Attributes

- [id](#) **sender**  
*Sender id.*
- [lava::key](#) **key**  
*Input key.*
- [lava::action](#) **action**  
*Input action.*
- [lava::mod](#) **mod**  
*Input mod.*
- [i32](#) **scancode** = 0  
*Input scan code.*



### 4.65.1 Detailed Description

Key event.

### 4.65.2 Member Function Documentation

#### 4.65.2.1 active()

```
bool lava::key_event::active () const [inline]
```

Check if key is active.

##### Returns

Key is pressed (and repeated) or not active

#### 4.65.2.2 pressed() [1/2]

```
bool lava::key_event::pressed (  
    key_ref k) const [inline]
```

Check if key is pressed.

##### Parameters

<i>k</i>	Key to check
----------	--------------

##### Returns

Key is pressed or not

#### 4.65.2.3 pressed() [2/2]

```
bool lava::key_event::pressed (  
    key_ref k,  
    mod_ref m) const [inline]
```

Check if key is pressed with mod.

##### Parameters

<i>k</i>	Key to check
<i>m</i>	Mods to check

##### Returns

Key is pressed with mod or not

#### 4.65.2.4 released()

```
bool lava::key_event::released (  
    key_ref k) const [inline]
```

Check if key is released.

**Parameters**

<i>k</i>	Key to check
----------	--------------

**Returns**

Key is released or not

**4.65.2.5 repeated()**

```
bool lava::key_event::repeated (  
    key\_ref k) const [inline]
```

Check if key is repeated.

**Parameters**

<i>k</i>	Key to check
----------	--------------

**Returns**

Key is repeated or not

The documentation for this struct was generated from the following file:

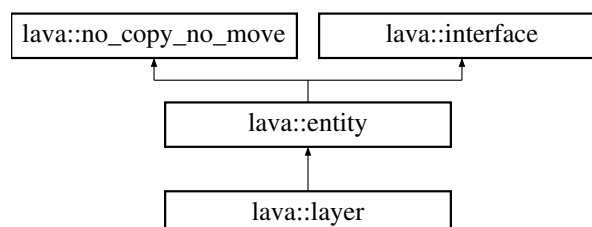
- [liblava/frame/input.hpp](#)

## 4.66 lava::layer Struct Reference

Layer.

```
#include <layer.hpp>
```

Inheritance diagram for lava::layer:



## Public Types

- using **s\_ptr** = std::shared\_ptr<layer>  
*Shared pointer to layer.*
- using **map** = std::map<id, s\_ptr>  
*Map of layers.*
- using **list** = std::vector<s\_ptr>  
*List of layers.*
- using **func** = std::function<void()>  
*Layer function.*

## Public Member Functions

- **layer** (string\_ref name)  
*Construct a new layer.*

## Public Member Functions inherited from lava::entity

- **entity** ()  
*Construct a new entity.*
- **id::ref get\_id** () const  
*Get the id of entity.*

## Public Member Functions inherited from lava::no\_copy\_no\_move

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Member Functions inherited from lava::interface

- virtual **~interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static **s\_ptr make** (string\_ref name)  
*Make a new layer.*

## Public Attributes

- **func on\_func**  
*Called by layering.*
- bool **active** = true  
*Active state.*
- **string name**  
*Name of layer.*

### 4.66.1 Detailed Description

Layer.

### 4.66.2 Constructor & Destructor Documentation

#### 4.66.2.1 layer()

```
lava::layer::layer (  
    string_ref name) [inline]
```

Construct a new layer.

##### Parameters

<i>name</i>	Name of layer
-------------	---------------

### 4.66.3 Member Function Documentation

#### 4.66.3.1 make()

```
static s_ptr lava::layer::make (  
    string_ref name) [inline], [static]
```

Make a new layer.

##### Parameters

<i>name</i>	Name of layer
-------------	---------------

##### Returns

s\_ptr Shared pointer to layer

The documentation for this struct was generated from the following file:

- liblava/util/[layer.hpp](#)

## 4.67 lava::texture::layer Struct Reference

Texture layer.

```
#include <texture.hpp>
```

## Public Types

- using **list** = std::vector<[layer](#)>  
*List of layers.*

## Public Attributes

- [mip\\_level::list](#) **levels**  
*List of mip levels.*

### 4.67.1 Detailed Description

Texture layer.

The documentation for this struct was generated from the following file:

- liblava/resource/[texture.hpp](#)

## 4.68 lava::layer\_list Struct Reference

Layer list.

```
#include <layer.hpp>
```

## Public Types

- using **ptr** = [layer\\_list](#)\*  
*Pointer to layer list.*

## Public Member Functions

- [id add](#) ([string\\_ref](#) name, [layer::func](#) func, bool active=true)  
*Add a new layer.*
- void [add](#) ([layer::s\\_ptr](#) layer)  
*Add a layer.*
- [id add\\_inactive](#) ([string\\_ref](#) name, [layer::func](#) func)  
*Add a new inactive layer.*
- [layer::s\\_ptr](#) [get](#) ([id::ref](#) layer\_id)  
*Get layer in list by id.*
- bool [remove](#) ([id::ref](#) layer\_id)  
*Remove layer from list.*
- [layer::list](#) const & [get\\_all](#) () const  
*Get all layers.*
- void **clear** ()  
*Clear layer list.*

### 4.68.1 Detailed Description

Layer list.

### 4.68.2 Member Function Documentation

#### 4.68.2.1 add() [1/2]

```
void lava::layer_list::add (  
    layer::s_ptr layer) [inline]
```

Add a layer.

##### Parameters

<i>layer</i>	Layer to add
--------------	--------------

#### 4.68.2.2 add() [2/2]

```
id lava::layer_list::add (  
    string_ref name,  
    layer::func func,  
    bool active = true) [inline]
```

Add a new layer.

##### Parameters

<i>name</i>	Name of layer
<i>func</i>	Layer function
<i>active</i>	Layer active state

##### Returns

id Id of added layer

#### 4.68.2.3 add\_inactive()

```
id lava::layer_list::add_inactive (  
    string_ref name,  
    layer::func func) [inline]
```

Add a new inactive layer.

##### Parameters

<i>name</i>	Name of layer
<i>func</i>	Layer function

##### Returns

id Id of added layer

#### 4.68.2.4 get()

```
layer::s_ptr lava::layer_list::get (  
    id::ref layer_id) [inline]
```

Get layer in list by id.

##### Parameters

<i>layer</i> ↔ _id	Id of layer
-----------------------	-------------

##### Returns

layer::ptr Shared pointer to layer

#### 4.68.2.5 get\_all()

```
layer::list const & lava::layer_list::get_all () const [inline]
```

Get all layers.

##### Returns

[layer::list](#) const& List of layers

#### 4.68.2.6 remove()

```
bool lava::layer_list::remove (  
    id::ref layer_id) [inline]
```

Remove layer from list.

##### Parameters

<i>layer</i> ↔ _id	Id of layer to remove
-----------------------	-----------------------

##### Returns

Remove was successful or failed

The documentation for this struct was generated from the following file:

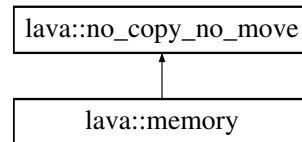
- [liblava/util/layer.hpp](#)

## 4.69 lava::memory Struct Reference

Vulkan memory.

```
#include <memory.hpp>
```

Inheritance diagram for lava::memory:



### Public Member Functions

- **memory** ()  
*Construct a new memory.*
- `VkAllocationCallbacks * alloc ()`  
*Get allocation callback.*
- `void set\_callbacks (VkAllocationCallbacks const &callbacks)`  
*Set the callbacks object.*
- `void set\_use\_custom\_cpu\_callbacks (bool value)`  
*Set use custom cpu callbacks.*

### Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- `void operator= (no\_copy\_no\_move const &)=delete`  
*No move.*

### Static Public Member Functions

- static [memory](#) & [instance](#) ()  
*Get memory instance.*

#### 4.69.1 Detailed Description

Vulkan memory.



## 4.69.2 Member Function Documentation

### 4.69.2.1 alloc()

```
VkAllocationCallbacks * lava::memory::alloc () [inline]
```

Get allocation callback.

#### Returns

VkAllocationCallbacks\* Allocation callbacks

### 4.69.2.2 instance()

```
static memory & lava::memory::instance () [inline], [static]
```

Get memory instance.

#### Returns

memory& Memory

### 4.69.2.3 set\_callbacks()

```
void lava::memory::set_callbacks (
    VkAllocationCallbacks const & callbacks) [inline]
```

Set the callbacks object.

#### Parameters

<i>callbacks</i>	Allocation Callbacks
------------------	----------------------

### 4.69.2.4 set\_use\_custom\_cpu\_callbacks()

```
void lava::memory::set_use_custom_cpu_callbacks (
    bool value) [inline]
```

Set use custom cpu callbacks.

#### Parameters

<i>value</i>	Value state
--------------	-------------

The documentation for this struct was generated from the following file:

- liblava/base/[memory.hpp](#)

## 4.70 lava::mesh\_meta Struct Reference

Mesh meta.

```
#include <mesh.hpp>
```

### Public Attributes

- **string filename**  
*Name of file (empty: see type)*
- **mesh\_type type** = mesh\_type::none  
*Mesh type.*

### 4.70.1 Detailed Description

Mesh meta.

The documentation for this struct was generated from the following file:

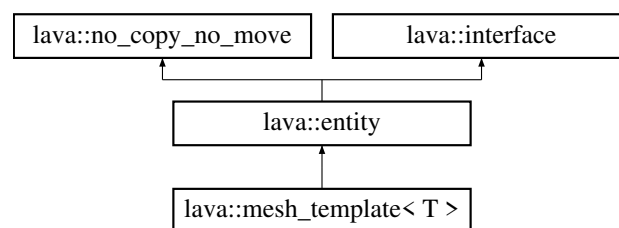
- liblava/resource/[mesh.hpp](#)

## 4.71 lava::mesh\_template< T > Struct Template Reference

Temporary templated mesh.

```
#include <mesh.hpp>
```

Inheritance diagram for lava::mesh\_template< T >:



### Public Types

- using **s\_ptr** = std::shared\_ptr<mesh\_template<T>>  
*Shared pointer to mesh.*
- using **s\_map** = std::map<id, s\_ptr>  
*Map of meshes.*
- using **s\_list** = std::vector<s\_ptr>  
*List of meshes.*
- using **vertex\_list** = std::vector<T>  
*List of vertices.*

**Public Member Functions**

- `~mesh_template ()`  
*Destroy the mesh.*
- `bool create (device::ptr device, bool mapped=false, VmaMemoryUsage memory_usage=VMA_MEMORY_USAGE_CPU_TO_GPU)`  
*Create a new mesh.*
- `void destroy ()`  
*Destroy the mesh.*
- `void bind (VkCommandBuffer cmd_buf) const`  
*Bind the mesh.*
- `void draw (VkCommandBuffer cmd_buf) const`  
*Draw the mesh.*
- `void bind_draw (VkCommandBuffer cmd_buf) const`  
*Bind and draw the mesh.*
- `bool empty () const`  
*Check if mesh is empty.*
- `void set_data (mesh_template_data< T > const &value)`  
*Set the mesh data.*
- `mesh_template_data< T > & get_data ()`  
*Get the mesh data.*
- `void add_data (mesh_template_data< T > const &value)`  
*Add mesh data to existing data.*
- `vertex_list & get_vertices ()`  
*Get the vertices of the mesh.*
- `vertex_list const & get_vertices () const`  
*Get the const vertices of the mesh.*
- `ui32 get_vertices_count () const`  
*Get the vertices count of the mesh.*
- `index_list & get_indices ()`  
*Get the indices of the mesh.*
- `index_list const & get_indices () const`  
*Get the const indices of the mesh.*
- `ui32 get_indices_count () const`  
*Get the indices count of the mesh.*
- `bool reload ()`  
*Reload the mesh data.*
- `buffer::s_ptr get_vertex_buffer ()`  
*Get the vertex buffer of the mesh.*
- `buffer::s_ptr get_index_buffer ()`  
*Get the index buffer of the mesh.*

**Public Member Functions inherited from `lava::entity`**

- `entity ()`  
*Construct a new entity.*
- `id::ref get_id () const`  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static [s\\_ptr](#) **make** ()  
*Make a new mesh.*

### 4.71.1 Detailed Description

```
template<typename T = vertex>
struct lava::mesh_template< T >
```

Temporary templated mesh.

Template Parameters

<i>T</i>	Vertex struct typename
----------	------------------------

### 4.71.2 Member Function Documentation

#### 4.71.2.1 add\_data()

```
template<typename T = vertex>
void lava::mesh_template< T >::add_data (
    mesh_template_data< T > const & value) [inline]
```

Add mesh data to existing data.

Parameters

<i>value</i>	Mesh data to add
--------------	------------------

#### 4.71.2.2 bind()

```
template<typename T >
void lava::mesh_template< T >::bind (
    VkCommandBuffer cmd_buf) const
```

Bind the mesh.

## Parameters

<i>cmd_buf</i>	Command buffer
----------------	----------------

## 4.71.2.3 bind\_draw()

```
template<typename T = vertex>
void lava::mesh_template< T >::bind_draw (
    VkCommandBuffer cmd_buf) const [inline]
```

Bind and draw the mesh.

## Parameters

<i>cmd_buf</i>	Command buffer
----------------	----------------

## 4.71.2.4 create()

```
template<typename T >
bool lava::mesh_template< T >::create (
    device::ptr device,
    bool mapped = false,
    VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_CPU_TO_GPU)
```

Create a new mesh.

## Parameters

<i>device</i>	Vulkan device
<i>mapped</i>	Map mesh data
<i>memory_usage</i>	Memory usage

## Returns

Create was successful or failed

## 4.71.2.5 draw()

```
template<typename T >
void lava::mesh_template< T >::draw (
    VkCommandBuffer cmd_buf) const
```

Draw the mesh.

## Parameters

<i>cmd_buf</i>	Command buffer
----------------	----------------

#### 4.71.2.6 empty()

```
template<typename T = vertex>
bool lava::mesh_template< T >::empty () const [inline]
```

Check if mesh is empty.

##### Returns

Mesh is empty or not

#### 4.71.2.7 get\_data()

```
template<typename T = vertex>
mesh_template_data< T > & lava::mesh_template< T >::get_data () [inline]
```

Get the mesh data.

##### Returns

mesh\_data& Mesh data

#### 4.71.2.8 get\_index\_buffer()

```
template<typename T = vertex>
buffer::s_ptr lava::mesh_template< T >::get_index_buffer () [inline]
```

Get the index buffer of the mesh.

##### Returns

buffer::s\_ptr Shared pointer to buffer

#### 4.71.2.9 get\_indices() [1/2]

```
template<typename T = vertex>
index_list & lava::mesh_template< T >::get_indices () [inline]
```

Get the indices of the mesh.

##### Returns

index\_list& List of indices

#### 4.71.2.10 `get_indices()` [2/2]

```
template<typename T = vertex>
index_list const & lava::mesh_template< T >::get_indices () const [inline]
```

Get the const indices of the mesh.

##### Returns

`index_list` const& List of indices

#### 4.71.2.11 `get_indices_count()`

```
template<typename T = vertex>
ui32 lava::mesh_template< T >::get_indices_count () const [inline]
```

Get the indices count of the mesh.

##### Returns

`ui32` Number of indices

#### 4.71.2.12 `get_vertex_buffer()`

```
template<typename T = vertex>
buffer::s_ptr lava::mesh_template< T >::get_vertex_buffer () [inline]
```

Get the vertex buffer of the mesh.

##### Returns

`buffer::s_ptr` Shared pointer to buffer

#### 4.71.2.13 `get_vertices()` [1/2]

```
template<typename T = vertex>
vertex_list & lava::mesh_template< T >::get_vertices () [inline]
```

Get the vertices of the mesh.

##### Returns

`vertex::list`& List of vertices

#### 4.71.2.14 get\_vertices() [2/2]

```
template<typename T = vertex>
vertex_list const & lava::mesh_template< T >::get_vertices () const [inline]
```

Get the const vertices of the mesh.

##### Returns

vertex::list const& List of vertices

#### 4.71.2.15 get\_vertices\_count()

```
template<typename T = vertex>
ui32 lava::mesh_template< T >::get_vertices_count () const [inline]
```

Get the vertices count of the mesh.

##### Returns

ui32 Number of vertices

#### 4.71.2.16 make()

```
template<typename T = vertex>
static s_ptr lava::mesh_template< T >::make () [inline], [static]
```

Make a new mesh.

##### Returns

s\_ptr Shared pointer to mesh

#### 4.71.2.17 reload()

```
template<typename T >
bool lava::mesh_template< T >::reload ()
```

Reload the mesh data.

##### Returns

Reload was successful or failed

#### 4.71.2.18 set\_data()

```
template<typename T = vertex>
void lava::mesh_template< T >::set_data (
    mesh_template_data< T > const & value) [inline]
```

Set the mesh data.



## Parameters

<i>value</i>	Mesh data
--------------	-----------

The documentation for this struct was generated from the following file:

- [liblava/resource/mesh.hpp](#)

## 4.72 `lava::mesh_template_data< T >` Struct Template Reference

Templated mesh data.

```
#include <mesh.hpp>
```

### Public Member Functions

- `template<typename PosType = r32>`  
`void move (std::array< PosType, 3 > offset)`  
*Move mesh data by offset.*
- `void scale (auto factor)`  
*Scale mesh data by factor.*
- `template<typename PosType = r32>`  
`void scale\_vector (std::array< PosType, 3 > factors)`  
*Scale mesh data by vector.*

### Public Attributes

- `std::vector< T > vertices`  
*List of vertices.*
- `index\_list indices`  
*List of indices.*

### 4.72.1 Detailed Description

```
template<typename T = vertex>
struct lava::mesh_template_data< T >
```

Templated mesh data.

#### Template Parameters

<i>T</i>	Input vertex struct
----------	---------------------

### 4.72.2 Member Function Documentation

#### 4.72.2.1 `move()`

```
template<typename T = vertex>
template<typename PosType = r32>
void lava::mesh\_template\_data< T >::move (
    std::array< PosType, 3 > offset) [inline]
```

Move mesh data by offset.

## Template Parameters

<i>PosType</i>	Coordinate element typename
----------------	-----------------------------

## Parameters

<i>offset</i>	Position offset
---------------	-----------------

**4.72.2.2 scale()**

```
template<typename T = vertex>
void lava::mesh_template_data< T >::scale (
    auto factor) [inline]
```

Scale mesh data by factor.

## Parameters

<i>factor</i>	Position scaling factor
---------------	-------------------------

**4.72.2.3 scale\_vector()**

```
template<typename T = vertex>
template<typename PosType = r32>
void lava::mesh_template_data< T >::scale_vector (
    std::array< PosType, 3 > factors) [inline]
```

Scale mesh data by vector.

## Template Parameters

<i>PosType</i>	Coordinate element typename
----------------	-----------------------------

## Parameters

<i>factors</i>	Array of position scaling factors
----------------	-----------------------------------

The documentation for this struct was generated from the following file:

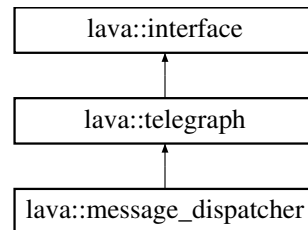
- [liblava/resource/mesh.hpp](#)

## 4.73 `lava::message_dispatcher` Struct Reference

Message dispatcher.

```
#include <telegram.hpp>
```

Inheritance diagram for `lava::message_dispatcher`:



### Public Types

- using **message\_func** = `std::function<void(telegram::ref, id::ref)>`  
*Message function.*

### Public Member Functions

- `~message_dispatcher ()`  
*Destroy the dispatcher.*
- `void setup (ui32 thread_count)`  
*Set up the dispatcher.*
- `void teardown ()`  
*Tear down the dispatcher.*
- `void update (ms current)`  
*Update the dispatcher.*
- `void send_message (id::ref receiver, id::ref sender, index message, ms delay={}, any const &info={})` override
- `bool add_dispatch (id::ref target, message_func func)`  
*Add dispatch.*
- `bool remove_dispatch (id::ref target)`  
*Remove dispatch.*
- `bool has_dispatch (id::ref target) const`  
*Check if dispatch is registered.*

### Public Member Functions inherited from `lava::telegraph`

### Public Member Functions inherited from `lava::interface`

- `virtual ~interface ()=default`  
*Destroy the interface.*

#### 4.73.1 Detailed Description

Message dispatcher.

## 4.73.2 Member Function Documentation

### 4.73.2.1 add\_dispatch()

```
bool lava::message_dispatcher::add_dispatch (
    id::ref target,
    message_func func) [inline]
```

Add dispatch.

#### Parameters

<i>target</i>	Sender id
<i>func</i>	Dispatch function

#### Returns

Dispatch added or not

### 4.73.2.2 has\_dispatch()

```
bool lava::message_dispatcher::has_dispatch (
    id::ref target) const [inline]
```

Check if dispatch is registered.

#### Parameters

<i>target</i>	Sender id
---------------	-----------

#### Returns

Dispatch exists or not

### 4.73.2.3 remove\_dispatch()

```
bool lava::message_dispatcher::remove_dispatch (
    id::ref target) [inline]
```

Remove dispatch.

#### Parameters

<i>target</i>	Sender id
---------------	-----------

#### Returns

Dispatch removed or not

#### 4.73.2.4 send\_message()

```
void lava::message_dispatcher::send_message (
    id::ref receiver,
    id::ref sender,
    index message,
    ms delay = {},
    any const & info = {}) [inline], [override], [virtual]
```

See also

[telegraph::send\\_message](#)

Implements [lava::telegraph](#).

#### 4.73.2.5 setup()

```
void lava::message_dispatcher::setup (
    ui32 thread_count) [inline]
```

Set up the dispatcher.

Parameters

<i>thread_count</i>	Number of threads
---------------------	-------------------

#### 4.73.2.6 update()

```
void lava::message_dispatcher::update (
    ms current) [inline]
```

Update the dispatcher.

Parameters

<i>current</i>	Time in milliseconds
----------------	----------------------

The documentation for this struct was generated from the following file:

- liblava/util/[telegram.hpp](#)

## 4.74 lava::texture::mip\_level Struct Reference

Texture mip level.

```
#include <texture.hpp>
```

## Public Types

- using **list** = std::vector<[mip\\_level](#)>  
*List of mip levels.*

## Public Attributes

- [uv2](#) **extent** {}  
*Mip level extent.*
- [ui32](#) **size** = 0  
*Mip level size.*

### 4.74.1 Detailed Description

Texture mip level.

The documentation for this struct was generated from the following file:

- liblava/resource/[texture.hpp](#)

## 4.75 lava::mouse\_active\_event Struct Reference

Mouse active event.

```
#include <input.hpp>
```

## Public Types

- using **ref** = [mouse\\_active\\_event](#) const&  
*Reference to mouse active event.*
- using **func** = std::function<bool([ref](#))>  
*Mouse active event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of mouse active event listeners.*
- using **list** = std::vector<[mouse\\_active\\_event](#)>  
*List of mouse active events.*

## Public Attributes

- [id](#) **sender**  
*Sender id.*
- bool **active** = false  
*Active state.*

### 4.75.1 Detailed Description

Mouse active event.

The documentation for this struct was generated from the following file:

- liblava/frame/[input.hpp](#)

## 4.76 lava::mouse\_button\_event Struct Reference

Mouse button event.

```
#include <input.hpp>
```

### Public Types

- using **ref** = [mouse\\_button\\_event](#) const&  
*Reference to mouse button event.*
- using **func** = std::function<bool([ref](#))>  
*Mouse button event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of mouse button event listeners.*
- using **list** = std::vector<[mouse\\_button\\_event](#)>  
*List of mouse button events.*

### Public Member Functions

- bool [pressed](#) ([mouse\\_button\\_ref](#) b) const  
*Check if mouse button is pressed.*
- bool [released](#) ([mouse\\_button\\_ref](#) b) const  
*Check if mouse button is released.*

### Public Attributes

- [id](#) **sender**  
*Sender id.*
- [mouse\\_button](#) **button**  
*Input mouse button.*
- [lava::action](#) **action**  
*Input action.*
- [lava::mod](#) **mod**  
*Input mod.*

### 4.76.1 Detailed Description

Mouse button event.

## 4.76.2 Member Function Documentation

### 4.76.2.1 pressed()

```
bool lava::mouse_button_event::pressed (
    mouse\_button\_ref b) const [inline]
```

Check if mouse button is pressed.

**Parameters**

<i>b</i>	Mouse button to check
----------	-----------------------

**Returns**

Mouse button is pressed or not

**4.76.2.2 released()**

```
bool lava::mouse_button_event::released (
    mouse_button_ref b) const [inline]
```

Check if mouse button is released.

**Parameters**

<i>b</i>	Mouse button to check
----------	-----------------------

**Returns**

Mouse button is released or not

The documentation for this struct was generated from the following file:

- [liblava/frame/input.hpp](#)

**4.77 lava::mouse\_move\_event Struct Reference**

Mouse move event.

```
#include <input.hpp>
```

**Public Types**

- using **ref** = [mouse\\_move\\_event](#) const&  
*Reference to mouse move event.*
- using **func** = std::function<bool([ref](#))>  
*Mouse move event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of mouse move event listeners.*
- using **list** = std::vector<[mouse\\_move\\_event](#)>  
*List of mouse move events.*



### Public Attributes

- `id` `sender`  
*Sender id.*
- `mouse_position` `position`  
*Input mouse position.*

#### 4.77.1 Detailed Description

Mouse move event.

The documentation for this struct was generated from the following file:

- `liblava/frame/input.hpp`

## 4.78 `lava::mouse_position` Struct Reference

Input mouse position.

```
#include <input.hpp>
```

### Public Attributes

- `r64` `x` = 0.0  
*X position.*
- `r64` `y` = 0.0  
*Y position.*

#### 4.78.1 Detailed Description

Input mouse position.

The documentation for this struct was generated from the following file:

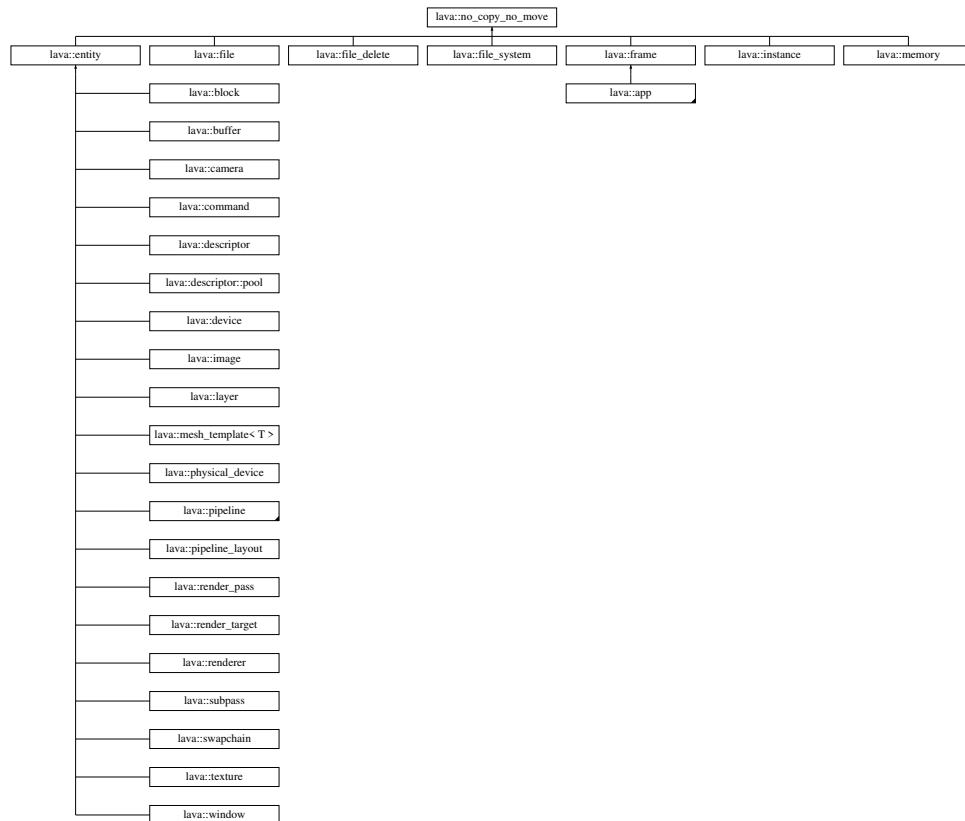
- `liblava/frame/input.hpp`

## 4.79 lava::no\_copy\_no\_move Struct Reference

No copy and no move object.

```
#include <types.hpp>
```

Inheritance diagram for lava::no\_copy\_no\_move:



### Public Member Functions

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

### 4.79.1 Detailed Description

No copy and no move object.

The documentation for this struct was generated from the following file:

- [liblava/core/types.hpp](#)

## 4.80 lava::pair\_hash Struct Reference

Pair hash.

```
#include <types.hpp>
```

### Public Member Functions

- `template<class T1 , class T2 >`  
`size_t operator() (std::pair< T1, T2 > const &p) const`  
*Hash operator.*

### 4.80.1 Detailed Description

Pair hash.

### 4.80.2 Member Function Documentation

#### 4.80.2.1 operator()()

```
template<class T1 , class T2 >  
size_t lava::pair_hash::operator() (  
    std::pair< T1, T2 > const & p) const [inline]
```

Hash operator.

#### Template Parameters

<i>T1</i>	Type of first
<i>T2</i>	Type of second

#### Parameters

<i>p</i>	Hash pair
----------	-----------

#### Returns

size\_t Hash value

The documentation for this struct was generated from the following file:

- liblava/core/[types.hpp](#)

## 4.81 lava::path\_drop\_event Struct Reference

Path drop event.

```
#include <input.hpp>
```

## Public Types

- using **ref** = [path\\_drop\\_event](#) const&  
*Reference to path drop event.*
- using **func** = std::function<bool([ref](#))>  
*Path drop event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of path drop event listeners.*
- using **list** = std::vector<[path\\_drop\\_event](#)>  
*List of path drop events.*

## Public Attributes

- [id](#) **sender**  
*Sender id.*
- [string\\_list](#) **files**  
*List of files.*

### 4.81.1 Detailed Description

Path drop event.

The documentation for this struct was generated from the following file:

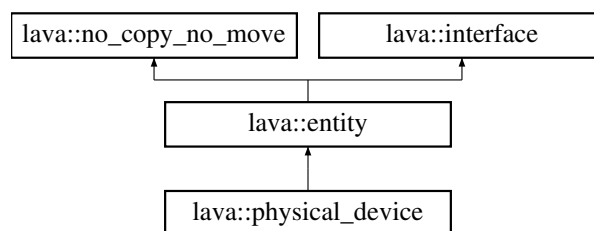
- [liblava/frame/input.hpp](#)

## 4.82 lava::physical\_device Struct Reference

Vulkan physical device.

```
#include <physical_device.hpp>
```

Inheritance diagram for lava::physical\_device:



## Public Types

- using **s\_ptr** = std::shared\_ptr<[physical\\_device](#)>  
*Shared pointer to physical device.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of physical devices.*
- using **ref** = [physical\\_device](#) const&  
*Reference to physical device.*

**Public Member Functions**

- **physical\_device** ()=default  
*Construct a new physical device.*
- **physical\_device** (VkPhysicalDevice vk\_physical\_device)  
*Construct and initialize a new physical device.*
- void **initialize** (VkPhysicalDevice vk\_physical\_device)  
*Initialize the physical device.*
- bool **supported** (string\_ref extension) const  
*Check if extension is supported.*
- bool **get\_queue\_family** (index &index, VkQueueFlags flags) const  
*Get the queue family.*
- **device::create\_param create\_default\_device\_param** () const  
*Create default device parameters.*
- VkPhysicalDeviceProperties const & **get\_properties** () const  
*Get the properties.*
- VkPhysicalDeviceFeatures const & **get\_features** () const  
*Get the features.*
- VkPhysicalDeviceMemoryProperties const & **get\_memory\_properties** () const  
*Get the memory properties.*
- **VkQueueFamilyPropertiesList** const & **get\_queue\_family\_properties** () const  
*Get the queue family properties.*
- **VkExtensionPropertiesList** const & **get\_extension\_properties** () const  
*Get the extension properties.*
- VkPhysicalDevice **get** () const  
*Get the Vulkan physical device.*
- **name get\_device\_name** () const  
*Get the device name.*
- **string get\_device\_type\_string** () const  
*Get the device type as string.*
- **sem\_version get\_driver\_version** () const  
*Get the driver version.*
- bool **swapchain\_supported** () const  
*Check if swapchain is supported.*
- bool **surface\_supported** (index queue\_family, VkSurfaceKHR surface) const  
*Check if surface is supported.*

**Public Member Functions inherited from [lava::entity](#)**

- **entity** ()  
*Construct a new entity.*
- **id::ref get\_id** () const  
*Get the id of entity.*

**Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)**

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default  
*Destroy the interface.*

## Static Public Member Functions

- static `s_ptr make` (VkPhysicalDevice vk\_physical\_device)  
*Make a new physical device.*

### 4.82.1 Detailed Description

Vulkan physical device.

### 4.82.2 Constructor & Destructor Documentation

#### 4.82.2.1 `physical_device()`

```
lava::physical_device::physical_device (
    VkPhysicalDevice vk_physical_device)
```

Construct and initialize a new physical device.

Parameters

<code>vk_physical_device</code>	Vulkan physical device
---------------------------------	------------------------

### 4.82.3 Member Function Documentation

#### 4.82.3.1 `create_default_device_param()`

```
device::create_param lava::physical_device::create_default_device_param () const
```

Create default device parameters.

Returns

`device::create_param` Device create parameters

#### 4.82.3.2 `get()`

```
VkPhysicalDevice lava::physical_device::get () const [inline]
```

Get the Vulkan physical device.

Returns

VkPhysicalDevice Vulkan physical device

#### 4.82.3.3 get\_device\_name()

```
name lava::physical_device::get_device_name () const
```

Get the device name.

##### Returns

name Name of device

#### 4.82.3.4 get\_device\_type\_string()

```
string lava::physical_device::get_device_type_string () const
```

Get the device type as string.

##### Returns

string String representation of device type

#### 4.82.3.5 get\_driver\_version()

```
sem_version lava::physical_device::get_driver_version () const
```

Get the driver version.

##### Returns

sem\_version Driver version

#### 4.82.3.6 get\_extension\_properties()

```
VkExtensionPropertiesList const & lava::physical_device::get_extension_properties () const  
[inline]
```

Get the extension properties.

##### Returns

VkExtensionPropertiesList const& List of extension properties

#### 4.82.3.7 get\_features()

```
VkPhysicalDeviceFeatures const & lava::physical_device::get_features () const [inline]
```

Get the features.

##### Returns

VkPhysicalDeviceFeatures const& Physical device features

**4.82.3.8 get\_memory\_properties()**

```
VkPhysicalDeviceMemoryProperties const & lava::physical_device::get_memory_properties () const
[inline]
```

Get the memory properties.

**Returns**

VkPhysicalDeviceMemoryProperties const& Physical device memory properties

**4.82.3.9 get\_properties()**

```
VkPhysicalDeviceProperties const & lava::physical_device::get_properties () const [inline]
```

Get the properties.

**Returns**

VkPhysicalDeviceProperties const& Physical device properties

**4.82.3.10 get\_queue\_family()**

```
bool lava::physical_device::get_queue_family (
    index & index,
    VkQueueFlags flags) const
```

Get the queue family.

**Parameters**

<i>index</i>	Returned index of queue family
<i>flags</i>	Queue flags that must be set

**Returns**

Found a queue family or not

**4.82.3.11 get\_queue\_family\_properties()**

```
VkQueueFamilyPropertiesList const & lava::physical_device::get_queue_family_properties ()
const [inline]
```

Get the queue family properties.

**Returns**

VkQueueFamilyPropertiesList const& List of queue family properties

**4.82.3.12 initialize()**

```
void lava::physical_device::initialize (
    VkPhysicalDevice vk_physical_device)
```

Initialize the physical device.



## Parameters

<i>vk_physical_device</i>	Vulkan physical device
---------------------------	------------------------

**4.82.3.13 make()**

```
static s_ptr lava::physical_device::make (  
    VkPhysicalDevice vk_physical_device) [inline], [static]
```

Make a new physical device.

## Parameters

<i>vk_physical_device</i>	Vulkan physical device
---------------------------	------------------------

## Returns

s\_ptr Shared pointer to physical device

**4.82.3.14 supported()**

```
bool lava::physical_device::supported (  
    string_ref extension) const
```

Check if extension is supported.

## Parameters

<i>extension</i>	Extension to check
------------------	--------------------

## Returns

Extension is supported or not

**4.82.3.15 surface\_supported()**

```
bool lava::physical_device::surface_supported (  
    index queue_family,  
    VkSurfaceKHR surface) const
```

Check if surface is supported.

## Parameters

<i>queue_family</i>	Index of queue family
<i>surface</i>	Vulkan surface

## Returns

Surface is supported or not

#### 4.82.3.16 swapchain\_supported()

```
bool lava::physical_device::swapchain_supported () const
```

Check if swapchain is supported.

##### Returns

Swapchain is supported or not

The documentation for this struct was generated from the following file:

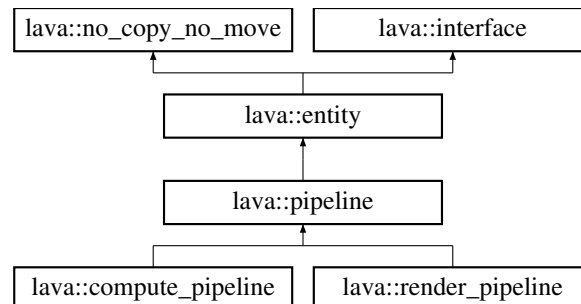
- liblava/base/[physical\\_device.hpp](#)

## 4.83 lava::pipeline Struct Reference

Pipeline.

```
#include <pipeline.hpp>
```

Inheritance diagram for lava::pipeline:



### Classes

- struct [shader\\_stage](#)  
*Shader stage.*

### Public Types

- using **s\_ptr** = std::shared\_ptr<[pipeline](#)>  
*Shared pointer to pipeline.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of pipelines.*
- using **process\_func** = std::function<void(VkCommandBuffer)>  
*Pipeline process function.*

**Public Member Functions**

- [pipeline](#) ([device::ptr device](#), VkPipelineCache pipeline\_cache=0)  
*Construct a new pipeline.*
- [~pipeline](#) () override  
*Destroy the pipeline.*
- bool [create](#) ()  
*Create a new pipeline.*
- void [destroy](#) ()  
*Destroy the pipeline.*
- virtual void [bind](#) (VkCommandBuffer cmd\_buf)=0  
*Bind the pipeline.*
- void [set\\_active](#) (bool value=true)  
*Set pipeline active.*
- bool [activated](#) () const  
*Check if pipeline is active.*
- void [toggle](#) ()  
*Toggle activation.*
- void [set\\_auto\\_bind](#) (bool value=true)  
*Set auto bind.*
- bool [auto\\_bind](#) () const  
*Check if auto bind is enabled.*
- bool [ready](#) () const  
*Check if pipeline is ready.*
- VkPipeline [get](#) () const  
*Get the pipeline.*
- [device::ptr](#) [get\\_device](#) ()  
*Get the device.*
- [pipeline\\_layout::s\\_ptr](#) [get\\_layout](#) () const  
*Get the layout.*
- void [set\\_layout](#) ([pipeline\\_layout::s\\_ptr](#) const &value)  
*Set the layout.*

**Public Member Functions inherited from [lava::entity](#)**

- [entity](#) ()  
*Construct a new entity.*
- [id::ref](#) [get\\_id](#) () const  
*Get the id of entity.*

**Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)**

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default  
*Destroy the interface.*

## Public Attributes

- `process_func` `on_process`  
*Called on pipeline process.*

## Protected Member Functions

- virtual bool `setup()`=0  
*Set up the pipeline.*
- virtual void `teardown()`=0  
*Tear down the pipeline.*

## Protected Attributes

- `device::ptr` `m_device` = nullptr  
*Vulkan device.*
- `VkPipeline` `m_vk_pipeline` = `VK_NULL_HANDLE`  
*Vulkan pipeline.*
- `VkPipelineCache` `m_pipeline_cache` = `VK_NULL_HANDLE`  
*Vulkan pipeline cache.*
- `pipeline_layout::s_ptr` `m_layout`  
*Pipeline layout.*

## 4.83.1 Detailed Description

Pipeline.

## 4.83.2 Constructor & Destructor Documentation

### 4.83.2.1 `pipeline()`

```
lava::pipeline::pipeline (
    device::ptr device,
    VkPipelineCache pipeline_cache = 0) [explicit]
```

Construct a new pipeline.

#### Parameters

<code>device</code>	Vulkan device
<code>pipeline_cache</code>	Pipeline cache

### 4.83.3 Member Function Documentation

#### 4.83.3.1 activated()

```
bool lava::pipeline::activated () const [inline]
```

Check if pipeline is active.

##### Returns

Pipeline is active or not

#### 4.83.3.2 auto\_bind()

```
bool lava::pipeline::auto_bind () const [inline]
```

Check if auto bind is enabled.

##### Returns

Auto bind is enabled or not

#### 4.83.3.3 bind()

```
virtual void lava::pipeline::bind (  
    VkCommandBuffer cmd_buf) [pure virtual]
```

Bind the pipeline.

##### Parameters

<i>cmd_buf</i>	Command buffer
----------------	----------------

Implemented in [lava::compute\\_pipeline](#), and [lava::render\\_pipeline](#).

#### 4.83.3.4 create()

```
bool lava::pipeline::create ()
```

Create a new pipeline.

##### Returns

Create was successful or failed

#### 4.83.3.5 get()

```
VkPipeline lava::pipeline::get () const [inline]
```

Get the pipeline.

Returns

VkPipeline Vulkan pipeline

#### 4.83.3.6 get\_device()

```
device::ptr lava::pipeline::get_device () [inline]
```

Get the device.

Returns

device::ptr Vulkan device

#### 4.83.3.7 get\_layout()

```
pipeline_layout::s_ptr lava::pipeline::get_layout () const [inline]
```

Get the layout.

Returns

pipeline\_layout::s\_ptr Pipeline layout

#### 4.83.3.8 ready()

```
bool lava::pipeline::ready () const [inline]
```

Check if pipeline is ready.

Returns

Pipeline is ready or not

#### 4.83.3.9 set\_active()

```
void lava::pipeline::set_active (  
    bool value = true) [inline]
```

Set pipeline active.

## Parameters

<i>value</i>	Active state
--------------	--------------

**4.83.3.10 set\_auto\_bind()**

```
void lava::pipeline::set_auto_bind (  
    bool value = true) [inline]
```

Set auto bind.

## Parameters

<i>value</i>	Enable state
--------------	--------------

**4.83.3.11 set\_layout()**

```
void lava::pipeline::set_layout (  
    pipeline_layout::s_ptr const & value) [inline]
```

Set the layout.

## Parameters

<i>value</i>	Pipeline layout
--------------	-----------------

**4.83.3.12 setup()**

```
virtual bool lava::pipeline::setup () [protected], [pure virtual]
```

Set up the pipeline.

## Returns

Setup was successful or failed

The documentation for this struct was generated from the following file:

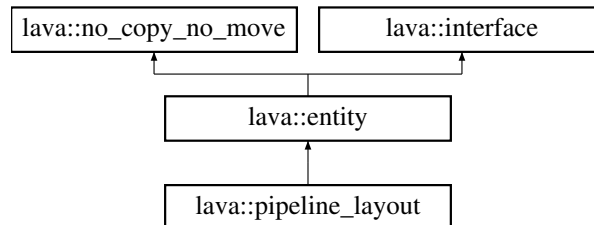
- liblava/block/[pipeline.hpp](#)

## 4.84 lava::pipeline\_layout Struct Reference

Pipeline layout.

```
#include <pipeline_layout.hpp>
```

Inheritance diagram for lava::pipeline\_layout:



### Public Types

- using **s\_ptr** = std::shared\_ptr<pipeline\_layout>  
*Shared pointer to pipeline layout.*
- using **s\_list** = std::vector<s\_ptr>  
*List of pipeline layouts.*
- using **offset\_list** = std::vector<index>  
*List of offsets.*

### Public Member Functions

- void **add** (descriptor::s\_ptr const &descriptor)  
*Add descriptor.*
- void **add** (VkPushConstantRange const &range)  
*Add push constant range.*
- void **add\_descriptor** (descriptor::s\_ptr const &descriptor)  
*Add descriptor.*
- void **add\_push\_constant\_range** (VkPushConstantRange const &range)  
*Add push constant range.*
- void **clear\_descriptors** ()  
*Clear descriptors.*
- void **clear\_ranges** ()  
*Clear push constant ranges.*
- void **clear** ()  
*Clear descriptors and push constant ranges.*
- bool **create** (device::ptr device)  
*Create a new pipeline layout.*
- void **destroy** ()  
*Destroy the pipeline layout.*
- VkPipelineLayout **get** () const  
*Get the Vulkan pipeline layout.*
- device::ptr **get\_device** ()  
*Get the device.*
- descriptor::s\_list const & **get\_descriptors** () const  
*Get the descriptors.*
- VkPushConstantRanges const & **get\_push\_constant\_ranges** () const  
*Get the push constant ranges.*
- void **bind\_descriptor\_set** (VkCommandBuffer cmd\_buf, VkDescriptorSet descriptor\_set, index first\_set=0, offset\_list offsets={}, VkPipelineBindPoint bind\_point=VK\_PIPELINE\_BIND\_POINT\_GRAPHICS)  
*Bind descriptor set.*
- void **bind** (VkCommandBuffer cmd\_buf, VkDescriptorSet descriptor\_set, index first\_set=0, offset\_list offsets={}, VkPipelineBindPoint bind\_point=VK\_PIPELINE\_BIND\_POINT\_GRAPHICS)



**Public Member Functions inherited from [lava::entity](#)**

- **entity** ()  
*Construct a new entity.*
- **id::ref** [get\\_id](#) () const  
*Get the id of entity.*

**Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)**

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

**Public Member Functions inherited from [lava::interface](#)**

- virtual **~interface** ()=default  
*Destroy the interface.*

**Static Public Member Functions**

- static [s\\_ptr](#) **make** ()  
*Make a new pipeline layout.*

**4.84.1 Detailed Description**

Pipeline layout.

**4.84.2 Member Function Documentation****4.84.2.1 [add\(\)](#) [1/2]**

```
void lava::pipeline_layout::add (
    descriptor::s\_ptr const & descriptor) [inline]
```

See also

[add\\_descriptor](#)

**4.84.2.2 [add\(\)](#) [2/2]**

```
void lava::pipeline_layout::add (
    VkPushConstantRange const & range) [inline]
```

See also

[add\\_push\\_constant\\_range](#)

**4.84.2.3 [add\\_descriptor\(\)](#)**

```
void lava::pipeline_layout::add_descriptor (
    descriptor::s\_ptr const & descriptor) [inline]
```

Add descriptor.

## Parameters

<i>descriptor</i>	Descriptor
-------------------	------------

**4.84.2.4 add\_push\_constant\_range()**

```
void lava::pipeline_layout::add_push_constant_range (
    VkPushConstantRange const & range) [inline]
```

Add push constant range.

## Parameters

<i>range</i>	Push constant range
--------------	---------------------

**4.84.2.5 bind()**

```
void lava::pipeline_layout::bind (
    VkCommandBuffer cmd_buf,
    VkDescriptorSet descriptor_set,
    index first_set = 0,
    offset_list offsets = {},
    VkPipelineBindPoint bind_point = VK_PIPELINE_BIND_POINT_GRAPHICS) [inline]
```

## See also

[bind\\_descriptor\\_set](#)

**4.84.2.6 bind\_descriptor\_set()**

```
void lava::pipeline_layout::bind_descriptor_set (
    VkCommandBuffer cmd_buf,
    VkDescriptorSet descriptor_set,
    index first_set = 0,
    offset_list offsets = {},
    VkPipelineBindPoint bind_point = VK_PIPELINE_BIND_POINT_GRAPHICS)
```

Bind descriptor set.

## Parameters

<i>cmd_buf</i>	Command buffer
<i>descriptor_set</i>	Descriptor set
<i>first_set</i>	Index to first descriptor set
<i>offsets</i>	List of offsets
<i>bind_point</i>	Pipeline bind point

**4.84.2.7 create()**

```
bool lava::pipeline_layout::create (
    device::ptr device)
```

Create a new pipeline layout.

## Parameters

<i>device</i>	Vulkan device
---------------	---------------

## Returns

Create was successful or failed

**4.84.2.8 get()**

```
VkPipelineLayout lava::pipeline_layout::get () const [inline]
```

Get the Vulkan pipeline layout.

## Returns

VkPipelineLayout Pipeline layout

**4.84.2.9 get\_descriptors()**

```
descriptor::s_list const & lava::pipeline_layout::get_descriptors () const [inline]
```

Get the descriptors.

## Returns

[descriptor::s\\_list](#) const& List of descriptors

**4.84.2.10 get\_device()**

```
device::ptr lava::pipeline_layout::get_device () [inline]
```

Get the device.

## Returns

[device::ptr](#) Vulkan device

**4.84.2.11 get\_push\_constant\_ranges()**

```
VkPushConstantRanges const & lava::pipeline_layout::get_push_constant_ranges () const [inline]
```

Get the push constant ranges.

## Returns

VkPushConstantRanges const& List of push constant ranges

#### 4.84.2.12 make()

```
static s_ptr lava::pipeline_layout::make () [inline], [static]
```

Make a new pipeline layout.

##### Returns

ptr Shared pointer to pipeline layout

The documentation for this struct was generated from the following file:

- liblava/block/pipeline\_layout.hpp

## 4.85 lava::platform Struct Reference

Stage platform.

```
#include <platform.hpp>
```

### Public Types

- using **ptr** = [platform\\*](#)  
*Pointer to platform.*
- using **create\_param\_func** = std::function<void([device::create\\_param&](#))>  
*Create parameter function.*

### Public Member Functions

- [device::s\\_ptr create](#) ([index physical\\_device=0](#))  
*Create a managed device from a physical device.*
- [device::s\\_ptr create](#) ([device::create\\_param::ref param](#))  
*Create a managed device with create parameters.*
- [device::ptr create\\_device](#) ([index physical\\_device=0](#))  
*Create a managed device.*
- [device::s\\_list](#) const & [get\\_devices](#) () const  
*Get all devices.*
- void **wait\_idle** ()  
*Wait for idle on all managed devices.*
- bool [remove](#) ([id::ref device\\_id](#))  
*Remove device from platform.*
- void **clear** ()  
*Clear all managed devices.*

### Public Attributes

- [create\\_param\\_func](#) **on\_create\_param**  
*Called on create to adjust the create parameters.*

### 4.85.1 Detailed Description

Stage platform.

### 4.85.2 Member Function Documentation

#### 4.85.2.1 create() [1/2]

```
device::s_ptr lava::platform::create (  
    device::create_param::ref param)
```

Create a managed device with create parameters.

##### Parameters

<i>param</i>	Create parameters
--------------	-------------------

##### Returns

*device::s\_ptr* Vulkan device

#### 4.85.2.2 create() [2/2]

```
device::s_ptr lava::platform::create (  
    index physical_device = 0)
```

Create a managed device from a physical device.

##### Parameters

<i>physical_device</i>	Index of physical device
------------------------	--------------------------

##### Returns

*device::s\_ptr* Vulkan device

#### 4.85.2.3 create\_device()

```
device::ptr lava::platform::create_device (  
    index physical_device = 0)
```

Create a managed device.

##### Parameters

<i>physical_device</i>	Physical device
------------------------	-----------------

##### Returns

*device::s\_ptr* Pointer to device

#### 4.85.2.4 get\_devices()

```
device::s_list const & lava::platform::get_devices () const [inline]
```

Get all devices.

Returns

`device::s_list` const& List of devices

#### 4.85.2.5 remove()

```
bool lava::platform::remove (
    id::ref device_id)
```

Remove device from platform.

Parameters

<code>device↔ _id</code>	Id of device
------------------------------	--------------

Returns

Remove was successful or failed

The documentation for this struct was generated from the following file:

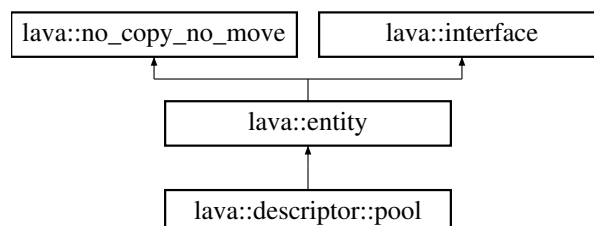
- [liblava/base/platform.hpp](#)

## 4.86 lava::descriptor::pool Struct Reference

Descriptor pool.

```
#include <descriptor.hpp>
```

Inheritance diagram for `lava::descriptor::pool`:



## Public Types

- using **s\_ptr** = std::shared\_ptr<pool>  
*Shared pointer to pool.*
- using **s\_list** = std::vector<s\_ptr>  
*List of pools.*

## Public Member Functions

- bool **create** (device::ptr device, VkDescriptorPoolSizesRef sizes, ui32 max=1, VkDescriptorPoolCreateFlags flags=VK\_DESCRIPTOR\_POOL\_CREATE\_FREE\_DESCRIPTOR\_SET\_BIT)  
*Create a new pool.*
- void **destroy** ()  
*Destroy the pool.*
- VkDescriptorPool **get** () const  
*Get the descriptor pool.*
- device::ptr **get\_device** ()  
*Get the device.*
- VkDescriptorPoolSizes const & **get\_sizes** () const  
*Get the sizes.*
- ui32 **get\_max** () const  
*Get the max.*

## Public Member Functions inherited from lava::entity

- **entity** ()  
*Construct a new entity.*
- id::ref **get\_id** () const  
*Get the id of entity.*

## Public Member Functions inherited from lava::no\_copy\_no\_move

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Member Functions inherited from lava::interface

- virtual ~**interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static s\_ptr **make** ()  
*Make a new descriptor pool.*

### 4.86.1 Detailed Description

Descriptor pool.

### 4.86.2 Member Function Documentation

#### 4.86.2.1 create()

```
bool lava::descriptor::pool::create (
    device::ptr device,
    VkDescriptorPoolSizesRef sizes,
    ui32 max = 1,
    VkDescriptorPoolCreateFlags flags = VK_DESCRIPTOR_POOL_CREATE_FREE_DESCRIPTOR_↵
    SET_BIT)
```

Create a new pool.

##### Parameters

<i>device</i>	Vulkan device
<i>sizes</i>	Descriptor pool sizes
<i>max</i>	Number of pools
<i>flags</i>	Create flags

##### Returns

Create was successful or failed

#### 4.86.2.2 get()

```
VkDescriptorPool lava::descriptor::pool::get () const [inline]
```

Get the descriptor pool.

##### Returns

VkDescriptorPool Vulkan descriptor pool

#### 4.86.2.3 get\_device()

```
device::ptr lava::descriptor::pool::get_device () [inline]
```

Get the device.

##### Returns

device::ptr Vulkan device



#### 4.86.2.4 get\_max()

```
ui32 lava::descriptor::pool::get_max () const [inline]
```

Get the max.

Returns

ui32 Number of pools

#### 4.86.2.5 get\_sizes()

```
VkDescriptorPoolSizes const & lava::descriptor::pool::get_sizes () const [inline]
```

Get the sizes.

Returns

VkDescriptorPoolSizes const& Descriptor pool sizes

#### 4.86.2.6 make()

```
static s_ptr lava::descriptor::pool::make () [inline], [static]
```

Make a new descriptor pool.

Returns

s\_ptr Shared pointer to descriptor pool

The documentation for this struct was generated from the following file:

- liblava/block/[descriptor.hpp](#)

## 4.87 lava::producer Struct Reference

Producer.

```
#include <producer.hpp>
```

### Public Types

- enum [shader\\_optimization](#) : index { **none** = 0 , **size** , **performance** }  
*Shader optimization level.*
- enum [shader\\_language](#) : index { **glsl** = 0 , **hlsl** }
- using **ptr** = [producer](#)\*
- *Pointer to producer.*

## Public Member Functions

- [mesh::s\\_ptr create\\_mesh](#) ([mesh\\_type](#) [mesh\\_type](#))  
*Create a mesh product.*
- [mesh::s\\_ptr get\\_mesh](#) ([string\\_ref](#) [name](#))  
*Get mesh by prop name.*
- [bool add\\_mesh](#) ([mesh::s\\_ptr](#) [mesh](#))  
*Add mesh to products.*
- [texture::s\\_ptr create\\_texture](#) ([uv2](#) [size](#))  
*Create a texture product.*
- [texture::s\\_ptr get\\_texture](#) ([string\\_ref](#) [name](#))  
*Get texture by prop name.*
- [bool add\\_texture](#) ([texture::s\\_ptr](#) [product](#))  
*Add texture to products.*
- [c\\_data get\\_shader](#) ([string\\_ref](#) [name](#), [bool](#) [reload=false](#))  
*Generate shader by prop name.*
- [c\\_data reload\\_shader](#) ([string\\_ref](#) [name](#))  
*Regenerate shader by prop name.*
- [data compile\\_shader](#) ([c\\_data](#) [product](#), [string\\_ref](#) [name](#), [string\\_ref](#) [filename](#)) [const](#)  
*Compile shader.*
- [void destroy](#) ()  
*Destroy all products.*
- [void clear](#) ()  
*Clear all products.*

## Public Attributes

- [engine](#) \* [app](#) = [nullptr](#)  
*Engine.*
- [id\\_registry](#)< [mesh](#), [string](#) > [meshes](#)  
*Mesh products.*
- [id\\_registry](#)< [texture](#), [string](#) > [textures](#)  
*Texture products.*
- [shader\\_optimization](#) [shader\\_opt](#) = [shader\\_optimization::performance](#)  
*Shader optimization level.*
- [shader\\_language](#) [shader\\_lang](#) = [shader\\_language::glsl](#)  
*Shader source language.*
- [bool](#) [shader\\_debug](#) = [false](#)  
*Shader debug information.*

### 4.87.1 Detailed Description

Producer.

### 4.87.2 Member Function Documentation

#### 4.87.2.1 add\_mesh()

```
bool lava::producer::add_mesh (
    mesh::s_ptr mesh)
```

Add mesh to products.

## Parameters

<i>mesh</i>	Mesh
-------------	------

## Returns

Added to products or already exists

### 4.87.2.2 add\_texture()

```
bool lava::producer::add_texture (  
    texture::s_ptr product)
```

Add texture to products.

## Parameters

<i>product</i>	Texture
----------------	---------

## Returns

Added to products or already exists

### 4.87.2.3 compile\_shader()

```
data lava::producer::compile_shader (  
    c_data product,  
    string_ref name,  
    string_ref filename) const
```

Compile shader.

## Parameters

<i>product</i>	Shader data
<i>name</i>	Shader name
<i>filename</i>	Shader filename

## Returns

data Compiled shader data

### 4.87.2.4 create\_mesh()

```
mesh::s_ptr lava::producer::create_mesh (  
    mesh_type mesh_type)
```

Create a mesh product.

## Parameters

<i>mesh_type</i>	Type of mesh
------------------	--------------

## Returns

[`mesh::s\_ptr`](#) Mesh

#### 4.87.2.5 create\_texture()

```
texture::s_ptr lava::producer::create_texture (
    uv2 size)
```

Create a texture product.

## Parameters

<i>size</i>	Size of texture
-------------	-----------------

## Returns

[`texture::s\_ptr`](#) Default texture

#### 4.87.2.6 get\_mesh()

```
mesh::s_ptr lava::producer::get_mesh (
    string_ref name)
```

Get mesh by prop name.

## Parameters

<i>name</i>	Name of prop
-------------	--------------

## Returns

[`mesh::s\_ptr`](#) Mesh

#### 4.87.2.7 get\_shader()

```
c_data lava::producer::get_shader (
    string_ref name,
    bool reload = false)
```

Generate shader by prop name.

## Parameters

<i>name</i>	Name of shader
<i>reload</i>	Reload shader

## Returns

[c\\_data](#) Shader data

**4.87.2.8 get\_texture()**

```
texture::s_ptr lava::producer::get_texture (  
    string_ref name)
```

Get texture by prop name.

## Parameters

<i>name</i>	Name of prop
-------------	--------------

## Returns

[texture::s\\_ptr](#) Texture

**4.87.2.9 reload\_shader()**

```
c_data lava::producer::reload_shader (  
    string_ref name) [inline]
```

Regenerate shader by prop name.

## Parameters

<i>name</i>	Name of shader
-------------	----------------

## Returns

[c\\_data](#) Shader data

The documentation for this struct was generated from the following file:

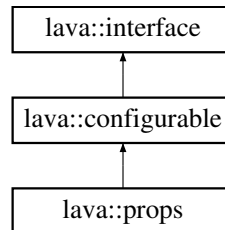
- [liblava/engine/producer.hpp](#)

## 4.88 lava::props Struct Reference

Props.

```
#include <props.hpp>
```

Inheritance diagram for lava::props:



### Classes

- struct [item](#)  
*Prop item.*

### Public Member Functions

- void [add](#) ([string\\_ref](#) name, [string\\_ref](#) filename)  
*Add a prop.*
- void [remove](#) ([string\\_ref](#) name)  
*Remove a prop.*
- bool [install](#) ([string\\_ref](#) name, [string\\_ref](#) filename)  
*Install a prop (add + load)*
- [c\\_data](#) operator() ([string\\_ref](#) name)  
*Get prop data.*
- [string\\_ref](#) [get\\_filename](#) ([string\\_ref](#) name) const  
*Get file name of prop.*
- void [set\\_filename](#) ([string\\_ref](#) name, [string\\_ref](#) filename)  
*Set filename of prop.*
- bool [exists](#) ([string\\_ref](#) name) const  
*Check if prop exists.*
- bool [empty](#) ([string\\_ref](#) name) const  
*Check if prop data is empty.*
- bool [load](#) ([string\\_ref](#) name)  
*Load prop data (reload if loaded)*
- void [unload](#) ([string\\_ref](#) name)  
*Unload prop data.*
- bool [load\\_all](#) ()  
*Load all prop data (reload if loaded)*
- void [unload\\_all](#) ()  
*Unload all prop data.*
- bool [check](#) ()  
*Check whether all props are available.*
- void [parse](#) ([cmd\\_line](#) cmd\_line)

- Parse prop overloads.*
- void **clear** ()  
*Clear all props.*
- [item::map](#) const & [get\\_all](#) () const  
*Get all props.*
- void [set\\_json](#) (json\_ref j) override
- json [get\\_json](#) () const override

## Public Member Functions inherited from [lava::configurable](#)

## Public Member Functions inherited from [lava::interface](#)

- virtual ~**interface** ()=default  
*Destroy the interface.*

## Public Attributes

- [engine](#) \* **app** = nullptr  
*Engine.*

## 4.88.1 Detailed Description

Props.

## 4.88.2 Member Function Documentation

### 4.88.2.1 add()

```
void lava::props::add (
    string_ref name,
    string_ref filename)
```

Add a prop.

#### Parameters

<i>name</i>	Name of prop
<i>filename</i>	File name of prop

### 4.88.2.2 check()

```
bool lava::props::check ()
```

Check whether all props are available.

#### Returns

All props are there or are missing (see log)

### 4.88.2.3 empty()

```
bool lava::props::empty (
    string_ref name) const [inline]
```

Check if prop data is empty.

**Parameters**

<i>name</i>	Name of prop
-------------	--------------

**Returns**

Prop data is empty or not

**4.88.2.4 exists()**

```
bool lava::props::exists (  
    string_ref name) const [inline]
```

Check if prop exists.

**Parameters**

<i>name</i>	Name of prop to check
-------------	-----------------------

**Returns**

Prop exists or not

**4.88.2.5 get\_all()**

```
item::map const & lava::props::get_all () const [inline]
```

Get all props.

**Returns**

[item::map](#) const& Map of props

**4.88.2.6 get\_filename()**

```
string_ref lava::props::get_filename (  
    string_ref name) const [inline]
```

Get file name of prop.

**Parameters**

<i>name</i>	Name of prop
-------------	--------------

**Returns**

string\_ref File name



#### 4.88.2.7 get\_json()

```
json lava::props::get_json () const [override], [virtual]
```

See also

[configurable::get\\_json](#)

Implements [lava::configurable](#).

#### 4.88.2.8 install()

```
bool lava::props::install (
    string_ref name,
    string_ref filename)
```

Install a prop (add + load)

Parameters

<i>name</i>	Name of prop
<i>filename</i>	File name of prop

Returns

Install was successful or failed

#### 4.88.2.9 load()

```
bool lava::props::load (
    string_ref name)
```

Load prop data (reload if loaded)

Parameters

<i>name</i>	Name of prop
-------------	--------------

Returns

Load was successful or failed

#### 4.88.2.10 load\_all()

```
bool lava::props::load_all ()
```

Load all prop data (reload if loaded)

Returns

Load was successful or failed

#### 4.88.2.11 operator>()()

```
c_data lava::props::operator() (
    string_ref name)
```

Get prop data.

## Parameters

<i>name</i>	Name of prop
-------------	--------------

## Returns

[c\\_data](#) Prop const data

**4.88.2.12 parse()**

```
void lava::props::parse (  
    cmd\_line cmd_line)
```

Parse prop overloads.

## Parameters

<i>cmd_line</i>	Command line arguments
-----------------	------------------------

**4.88.2.13 remove()**

```
void lava::props::remove (  
    string\_ref name)
```

Remove a prop.

## Parameters

<i>name</i>	Name of prop
-------------	--------------

**4.88.2.14 set\_filename()**

```
void lava::props::set_filename (  
    string\_ref name,  
    string\_ref filename) [inline]
```

Set filename of prop.

## Parameters

<i>name</i>	Name of prop
<i>filename</i>	File name

#### 4.88.2.15 set\_json()

```
void lava::props::set_json (
    json_ref j)  [override], [virtual]
```

See also

[configurable::set\\_json](#)

Implements [lava::configurable](#).

#### 4.88.2.16 unload()

```
void lava::props::unload (
    string_ref name)  [inline]
```

Unload prop data.

Parameters

<i>name</i>	Name of prop
-------------	--------------

The documentation for this struct was generated from the following file:

- [liblava/engine/props.hpp](#)

## 4.89 lava::pseudorandom\_generator Struct Reference

Pseudorandom generator.

```
#include <random.hpp>
```

### Public Member Functions

- [pseudorandom\\_generator](#) ([ui32](#) seed)  
*Construct a new pseudorandom generator.*
- void [set\\_seed](#) ([ui32](#) value)  
*Set the seed.*
- [ui32](#) [get](#) ()  
*Get next pseudorandom number.*

### 4.89.1 Detailed Description

Pseudorandom generator.

### 4.89.2 Constructor & Destructor Documentation

#### 4.89.2.1 pseudorandom\_generator()

```
lava::pseudorandom_generator::pseudorandom_generator (
    ui32 seed)  [inline], [explicit]
```

Construct a new pseudorandom generator.

## Parameters

<i>seed</i>	Seed for generator
-------------	--------------------

### 4.89.3 Member Function Documentation

#### 4.89.3.1 `get()`

```
ui32 lava::pseudorandom_generator::get () [inline]
```

Get next pseudorandom number.

## Returns

ui32 Random number

#### 4.89.3.2 `set_seed()`

```
void lava::pseudorandom_generator::set_seed (
    ui32 value) [inline]
```

Set the seed.

## Parameters

<i>value</i>	Generator seed
--------------	----------------

The documentation for this struct was generated from the following file:

- [liblava/util/random.hpp](#)

## 4.90 `lava::queue` Struct Reference

Device queue.

```
#include <queue.hpp>
```

## Public Types

- using **list** = std::deque<[queue](#)>  
*List of queues.*
- using **ref** = [queue](#) const&  
*Reference to queue.*

## Public Member Functions

- bool [valid](#) () const  
*Check if queue is valid.*
- bool [operator<](#) ([queue](#) const &[other](#)) const  
*Queue priority compare operator.*

## Public Attributes

- VkQueue **vk\_queue** = nullptr  
*Vulkan queue.*
- VkQueueFlags **flags** = 0  
*Queue flags.*
- [index](#) **family** = 0  
*Queue family index.*
- [r32](#) **priority** = 1.f  
*Queue priority.*

### 4.90.1 Detailed Description

Device queue.

### 4.90.2 Member Function Documentation

#### 4.90.2.1 [operator<\(\)](#)

```
bool lava::queue::operator< (
    queue const & other) const [inline]
```

Queue priority compare operator.

#### Parameters

<i>other</i>	Queue to compare
--------------	------------------

#### Returns

Priority of queue is higher or lower and equal

#### 4.90.2.2 [valid\(\)](#)

```
bool lava::queue::valid () const [inline]
```

Check if queue is valid.

#### Returns

Queue is valid or not

The documentation for this struct was generated from the following file:

- liblava/base/[queue.hpp](#)

## 4.91 `lava::queue_family_info` Struct Reference

Queue family information.

```
#include <queue.hpp>
```

### Public Types

- using **list** = `std::deque<queue\_family\_info>`  
*List of queue family informations.*

### Public Member Functions

- void **add** (`VkQueueFlags` flags, `ui32` count=1, `r32` priority=1.f)  
*Add a queue family information.*
- `ui32` **count** () const  
*Get the count of queues.*
- void **clear** ()  
*Clear the queue information.*

### Public Attributes

- `index` **family\_index** = 0  
*Queue family index.*
- `queue_info::list` **queues**  
*List of queue informations.*

### 4.91.1 Detailed Description

Queue family information.

### 4.91.2 Member Function Documentation

#### 4.91.2.1 `add()`

```
void lava::queue_family_info::add (  
    VkQueueFlags flags,  
    ui32 count = 1,  
    r32 priority = 1.f) [inline]
```

Add a queue family information.

#### Parameters

<i>flags</i>	Queue flags
<i>count</i>	Number of queues
<i>priority</i>	Queue priority

#### 4.91.2.2 count()

```
ui32 lava::queue_family_info::count () const [inline]
```

Get the count of queues.

##### Returns

ui32 Count of queues

The documentation for this struct was generated from the following file:

- [liblava/base/queue.hpp](#)

## 4.92 lava::queue\_info Struct Reference

Queue information.

```
#include <queue.hpp>
```

### Public Types

- using **list** = std::deque<[queue\\_info](#)>  
*List of queue informations.*

### Public Attributes

- VkQueueFlags **flags** = [default\\_queue\\_flags](#)  
*Queue flags.*
- [r32](#) **priority** = 1.f  
*Queue priority.*

### 4.92.1 Detailed Description

Queue information.

The documentation for this struct was generated from the following file:

- [liblava/base/queue.hpp](#)

## 4.93 lava::random\_generator Struct Reference

Random generator.

```
#include <random.hpp>
```

## Public Member Functions

- **random\_generator** ()  
*Construct a new random generator.*
- **i32 get** (i32 low, i32 high)  
*Get next random number.*
- **template<typename T = real>**  
**T get** (T low, T high)  
*Get next real random number.*

### 4.93.1 Detailed Description

Random generator.

### 4.93.2 Member Function Documentation

#### 4.93.2.1 get() [1/2]

```
i32 lava::random_generator::get (
    i32 low,
    i32 high) [inline]
```

Get next random number.

#### Parameters

<i>low</i>	Lowest number
<i>high</i>	Highest number

#### Returns

i32 Random number

#### 4.93.2.2 get() [2/2]

```
template<typename T = real>
T lava::random_generator::get (
    T low,
    T high) [inline]
```

Get next real random number.

#### Template Parameters

<i>T</i>	Type of number
----------	----------------



## Parameters

<i>low</i>	Lowest number
<i>high</i>	Highest number

## Returns

T Random number

The documentation for this struct was generated from the following file:

- `liblava/util/random.hpp`

## 4.94 `lava::rect` Struct Reference

Rectangle.

```
#include <math.hpp>
```

## Public Types

- using **ref** = `rect` const&  
*Reference to rect.*

## Public Member Functions

- **rect** ()=default  
*Construct a new rectangle.*
- **rect** (i32 left, i32 top, ui32 width, ui32 height)  
*Construct a new rectangle.*
- **rect** (iv2 const &left\_top, ui32 width, ui32 height)  
*Construct a new rectangle.*
- **rect** (iv2 const &left\_top, uv2 const &size)  
*Construct a new rectangle.*
- iv2 const & **get\_origin** () const  
*Get the origin.*
- iv2 const & **get\_end\_point** () const  
*Get the end point.*
- uv2 **get\_size** () const  
*Get the size.*
- void **set\_size** (uv2 const &size)  
*Set the size.*
- void **move** (iv2 const &offset)  
*Move the rectangle.*
- bool **contains** (iv2 point) const  
*Check if point is inside the rectangle.*

### 4.94.1 Detailed Description

Rectangle.

### 4.94.2 Constructor & Destructor Documentation

#### 4.94.2.1 `rect()` [1/3]

```
lava::rect::rect (  
    i32 left,  
    i32 top,  
    ui32 width,  
    ui32 height) [inline]
```

Construct a new rectangle.

##### Parameters

<i>left</i>	Left position
<i>top</i>	Top position
<i>width</i>	Rectangle width
<i>height</i>	Rectangle height

#### 4.94.2.2 `rect()` [2/3]

```
lava::rect::rect (  
    iv2 const & left_top,  
    ui32 width,  
    ui32 height) [inline]
```

Construct a new rectangle.

##### Parameters

<i>left_top</i>	Left top position
<i>width</i>	Rectangle width
<i>height</i>	Rectangle height

#### 4.94.2.3 `rect()` [3/3]

```
lava::rect::rect (  
    iv2 const & left_top,  
    uv2 const & size) [inline]
```

Construct a new rectangle.

## Parameters

<i>left_top</i>	Left top position
<i>size</i>	Size of rectangle

### 4.94.3 Member Function Documentation

#### 4.94.3.1 contains()

```
bool lava::rect::contains (  
    iv2 point) const [inline]
```

Check if point is inside the rectangle.

## Parameters

<i>point</i>	Point to check
--------------	----------------

## Returns

Point is inside or out

#### 4.94.3.2 get\_end\_point()

```
iv2 const & lava::rect::get_end_point () const [inline]
```

Get the end point.

## Returns

iv2 const& Right bottom position

#### 4.94.3.3 get\_origin()

```
iv2 const & lava::rect::get_origin () const [inline]
```

Get the origin.

## Returns

iv2 const& Left top position

#### 4.94.3.4 get\_size()

```
uv2 lava::rect::get_size () const [inline]
```

Get the size.

## Returns

uv2 Width and height

#### 4.94.3.5 move()

```
void lava::rect::move (  
    iv2 const & offset) [inline]
```

Move the rectangle.

## Parameters

<i>offset</i>	Offset to move
---------------	----------------

**4.94.3.6 set\_size()**

```
void lava::rect::set_size (
    uv2 const & size) [inline]
```

Set the size.

## Parameters

<i>size</i>	Width and height
-------------	------------------

The documentation for this struct was generated from the following file:

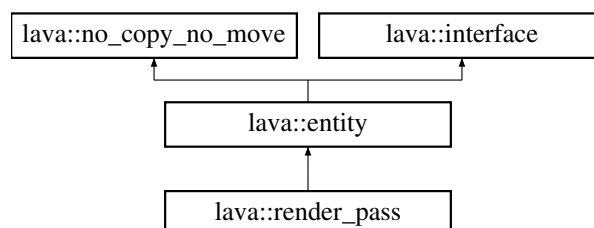
- liblava/util/[math.hpp](#)

**4.95 lava::render\_pass Struct Reference**

Render pass.

```
#include <render_pass.hpp>
```

Inheritance diagram for lava::render\_pass:

**Public Types**

- using **s\_ptr** = std::shared\_ptr<[render\\_pass](#)>  
*Shared pointer to render pass.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of render passes.*

## Public Member Functions

- [render\\_pass](#) ([device::ptr](#) device)  
*Construct a new render pass.*
- bool [create](#) ([VkAttachmentsRef](#) target\_attachments, [rect::ref](#) area)  
*Create a new render pass.*
- void **destroy** ()  
*Destroy the render pass.*
- void [process](#) ([VkCommandBuffer](#) cmd\_buf, [index](#) frame)  
*Process the render pass.*
- [device::ptr](#) [get\\_device](#) ()  
*Get the device.*
- [VkRenderPass](#) [get](#) () const  
*Get the render pass.*
- [ui32](#) [get\\_subpass\\_count](#) () const  
*Get the subpass count.*
- bool [exists\\_subpass](#) ([index](#) index=0) const  
*Check if subpass exists.*
- [subpass](#) \* [get\\_subpass](#) ([index](#) index=0)  
*Get the subpass.*
- [subpass::s\\_list](#) const & [get\\_subpasses](#) () const  
*Get the subpasses.*
- void [add](#) ([attachment::s\\_ptr](#) const &attachment)  
*Add an attachment.*
- void [add](#) ([subpass\\_dependency::s\\_ptr](#) const &dependency)  
*Add a subpass dependency.*
- void [add](#) ([subpass::s\\_ptr](#) const &subpass)  
*Add a subpass.*
- void [set\\_clear\\_values](#) ([VkClearValues](#) const &values)  
*Set the clear values.*
- [VkClearValues](#) const & [get\\_clear\\_values](#) () const  
*Get the clear values.*
- void [set\\_clear\\_color](#) ([v3](#) value={})  
*Set the clear color.*
- [v3](#) [get\\_clear\\_color](#) () const  
*Get the clear color.*
- void [add](#) ([render\\_pipeline::s\\_ptr](#) pipeline, [index](#) subpass=0)  
*Add a render pipeline to the back of subpass.*
- void [add\\_front](#) ([render\\_pipeline::s\\_ptr](#) pipeline, [index](#) subpass=0)  
*Add a render pipeline to the front of subpass.*
- void [remove](#) ([render\\_pipeline::s\\_ptr](#) pipeline, [index](#) subpass=0)  
*Remove a render pipeline from the subpass.*
- [target\\_callback](#) const & [get\\_target\\_callback](#) () const  
*Get the target callback.*

Public Member Functions inherited from [lava::entity](#)

- **entity** ()  
*Construct a new entity.*
- [id::ref](#) [get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static [s\\_ptr](#) **make** ([device::ptr](#) device)  
*Make a new render pass.*

### 4.95.1 Detailed Description

Render pass.

### 4.95.2 Constructor & Destructor Documentation

#### 4.95.2.1 render\_pass()

```
lava::render_pass::render_pass (
    device::ptr device) [explicit]
```

Construct a new render pass.

Parameters

<i>device</i>	Vulkan device
---------------	---------------

### 4.95.3 Member Function Documentation

#### 4.95.3.1 add() [1/4]

```
void lava::render_pass::add (
    attachment::s\_ptr const & attachment) [inline]
```

Add an attachment.

## Parameters

<i>attachment</i>	Attachment
-------------------	------------

## 4.95.3.2 add() [2/4]

```
void lava::render_pass::add (  
    render_pipeline::s_ptr pipeline,  
    index subpass = 0) [inline]
```

Add a render pipeline to the back of subpass.

## Parameters

<i>pipeline</i>	Render pipeline
<i>subpass</i>	Subpass

## 4.95.3.3 add() [3/4]

```
void lava::render_pass::add (  
    subpass::s_ptr const & subpass) [inline]
```

Add a subpass.

## Parameters

<i>subpass</i>	Subpass
----------------	---------

## 4.95.3.4 add() [4/4]

```
void lava::render_pass::add (  
    subpass_dependency::s_ptr const & dependency) [inline]
```

Add a subpass dependency.

## Parameters

<i>dependency</i>	Subpass dependency
-------------------	--------------------

## 4.95.3.5 add\_front()

```
void lava::render_pass::add_front (  
    render_pipeline::s_ptr pipeline,  
    index subpass = 0) [inline]
```

Add a render pipeline to the front of subpass.

## Parameters

<i>pipeline</i>	Render pipeline
<i>subpass</i>	Subpass

**4.95.3.6 create()**

```
bool lava::render_pass::create (  
    VkAttachmentsRef target_attachments,  
    rect::ref area)
```

Create a new render pass.

## Parameters

<i>target_attachments</i>	List of target attachments
<i>area</i>	Rectangle area

## Returns

Create was successful or failed

**4.95.3.7 exists\_subpass()**

```
bool lava::render_pass::exists_subpass (  
    index index = 0) const [inline]
```

Check if subpass exists.

## Parameters

<i>index</i>	Index to check
--------------	----------------

## Returns

Subpass exists or not

**4.95.3.8 get()**

```
VkRenderPass lava::render_pass::get () const [inline]
```

Get the render pass.

## Returns

VkRenderPass Vulkan render pass



#### 4.95.3.9 get\_clear\_color()

```
v3 lava::render_pass::get_clear_color () const
```

Get the clear color.

Returns

v3 Clear color

#### 4.95.3.10 get\_clear\_values()

```
VkClearValues const & lava::render_pass::get_clear_values () const [inline]
```

Get the clear values.

Returns

VkClearValues const& List of clear values

#### 4.95.3.11 get\_device()

```
device::ptr lava::render_pass::get_device () [inline]
```

Get the device.

Returns

device::ptr Vulkan device

#### 4.95.3.12 get\_subpass()

```
subpass * lava::render_pass::get_subpass (
    index index = 0) [inline]
```

Get the subpass.

Parameters

<i>index</i>	Index of subpass
--------------	------------------

Returns

subpass\* Subpass

#### 4.95.3.13 get\_subpass\_count()

```
ui32 lava::render_pass::get_subpass_count () const [inline]
```

Get the subpass count.

##### Returns

ui32 Number of subpasses

#### 4.95.3.14 get\_subpasses()

```
subpass::s_list const & lava::render_pass::get_subpasses () const [inline]
```

Get the subpasses.

##### Returns

subpass::s\_list const& List of subpasses

#### 4.95.3.15 get\_target\_callback()

```
target_callback const & lava::render_pass::get_target_callback () const [inline]
```

Get the target callback.

##### Returns

target\_callback const& Target callback

#### 4.95.3.16 make()

```
static s_ptr lava::render_pass::make (  
    device::ptr device) [inline], [static]
```

Make a new render pass.

##### Parameters

<i>device</i>	Vulkan device
---------------	---------------

##### Returns

s\_ptr Shared pointer to render pass

#### 4.95.3.17 process()

```
void lava::render_pass::process (  
    VkCommandBuffer cmd_buf,  
    index frame)
```

Process the render pass.

## Parameters

<i>cmd_buf</i>	Command buffer
<i>frame</i>	Frame index

**4.95.3.18 remove()**

```
void lava::render_pass::remove (
    render_pipeline::s_ptr pipeline,
    index subpass = 0) [inline]
```

Remove a render pipeline from the subpass.

## Parameters

<i>pipeline</i>	Render pipeline
<i>subpass</i>	Subpass

**4.95.3.19 set\_clear\_color()**

```
void lava::render_pass::set_clear_color (
    v3 value = {}) [inline]
```

Set the clear color.

## Parameters

<i>value</i>	Clear color
--------------	-------------

**4.95.3.20 set\_clear\_values()**

```
void lava::render_pass::set_clear_values (
    VkClearValues const & values) [inline]
```

Set the clear values.

## Parameters

<i>values</i>	List of clear values
---------------	----------------------

The documentation for this struct was generated from the following file:

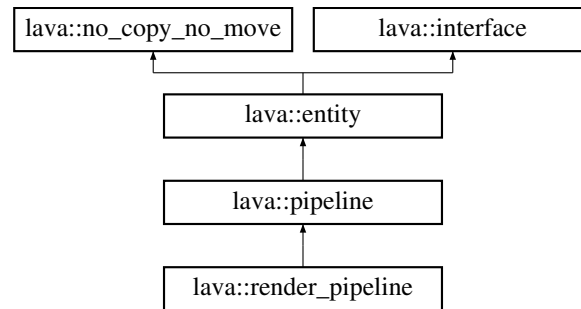
- liblava/block/[render\\_pass.hpp](#)

## 4.96 lava::render\_pipeline Struct Reference

Render pipeline (Graphics)

```
#include <render_pipeline.hpp>
```

Inheritance diagram for lava::render\_pipeline:



### Classes

- struct [create\\_info](#)  
*Render pipeline create information.*

### Public Types

- enum class [sizing\\_mode](#) : index { **input** = 0 , **absolute** , **relative** }  
*Sizing modes.*
- using **s\_ptr** = std::shared\_ptr<[render\\_pipeline](#)>  
*Shared pointer to render pipeline.*
- using **s\_map** = std::map<[id](#), **s\_ptr**>  
*Map of render pipelines.*
- using **s\_list** = std::vector<**s\_ptr**>  
*List of render pipelines.*
- using **create\_func** = std::function<bool([create\\_info](#)&)>  
*Create function.*

### Public Types inherited from [lava::pipeline](#)

- using **s\_ptr** = std::shared\_ptr<[pipeline](#)>  
*Shared pointer to pipeline.*
- using **s\_list** = std::vector<**s\_ptr**>  
*List of pipelines.*
- using **process\_func** = std::function<void(VkCommandBuffer)>  
*Pipeline process function.*

## Public Member Functions

- [render\\_pipeline](#) ([device::ptr device](#), [VkPipelineCache pipeline\\_cache](#))  
*Construct a new render pipeline.*
- void [bind](#) ([VkCommandBuffer cmd\\_buf](#)) override  
*Bind the pipeline.*
- void [set\\_viewport\\_and\\_scissor](#) ([VkCommandBuffer cmd\\_buf](#), [uv2 size](#))  
*Set the viewport and scissor.*
- void [set\\_render\\_pass](#) ([VkRenderPass pass](#))  
*Set the render pass.*
- void [set](#) ([VkRenderPass pass](#))
- [VkRenderPass get\\_render\\_pass](#) () const  
*Get the render pass.*
- [index get\\_subpass](#) () const  
*Get the subpass.*
- void [set\\_subpass](#) ([index value](#))  
*Set the subpass.*
- bool [create](#) ([VkRenderPass pass](#))  
*Create a new render pipeline.*
- void [set\\_vertex\\_input\\_binding](#) ([VkVertexInputBindingDescription const &description](#))  
*Set the vertex input binding.*
- void [set\\_vertex\\_input\\_bindings](#) ([VkVertexInputBindingDescriptions const &descriptions](#))  
*Set the vertex input bindings.*
- void [set\\_vertex\\_input\\_attribute](#) ([VkVertexInputAttributeDescription const &attribute](#))  
*Set the vertex input attribute.*
- void [set\\_vertex\\_input\\_attributes](#) ([VkVertexInputAttributeDescriptions const &attributes](#))  
*Set the vertex input attributes.*
- void [set\\_input\\_topology](#) ([VkPrimitiveTopology const &topology](#))  
*Set the input assembler's topology.*
- void [set\\_depth\\_test\\_and\\_write](#) (bool test\_enable=true, bool write\_enable=true)  
*Set the depth test and write.*
- void [set\\_depth\\_compare\\_op](#) ([VkCompareOp compare\\_op](#))  
*Set the depth compare operation.*
- void [set\\_rasterization\\_cull\\_mode](#) ([VkCullModeFlags cull\\_mode](#))  
*Set the rasterization cull mode.*
- void [set\\_rasterization\\_front\\_face](#) ([VkFrontFace front\\_face](#))  
*Set the rasterization front face.*
- void [set\\_rasterization\\_polygon\\_mode](#) ([VkPolygonMode polygon\\_mode](#))  
*Set the rasterization polygon mode.*
- void [add\\_color\\_blend\\_attachment](#) ([VkPipelineColorBlendAttachmentState const &attachment](#))  
*Add color blend attachment.*
- void [add\\_color\\_blend\\_attachment](#) ()  
*Add color blend attachment (default)*
- void [clear\\_color\\_blend\\_attachment](#) ()  
*Clear color blend attachment.*
- void [set\\_dynamic\\_states](#) ([VkDynamicStates const &states](#))  
*Set the dynamic states.*
- void [add\\_dynamic\\_state](#) ([VkDynamicState state](#))  
*Add a dynamic state.*
- void [clear\\_dynamic\\_states](#) ()  
*Clear dynamic states.*

- bool [add\\_shader\\_stage](#) (c\_data::ref data, VkShaderStageFlagBits [stage](#))  
*Add shader stage.*
- bool [add\\_shader](#) (c\_data::ref data, VkShaderStageFlagBits [stage](#))  
*Add shader.*
- void [add](#) (shader\_stage::s\_ptr const &[shader\\_stage](#))  
*Add shader stage.*
- shader\_stage::s\_list const & [get\\_shader\\_stages](#) () const  
*Get the shader stages.*
- void **clear\_shader\_stages** ()  
*Clear the shader stages.*
- void **clear** ()  
*Clear the render pipeline.*
- void [set\\_auto\\_size](#) (bool value=true)  
*Set the auto size.*
- bool [auto\\_sizing](#) () const  
*Get the auto sizing state.*
- VkViewport [get\\_viewport](#) () const  
*Get the viewport.*
- void [set\\_viewport](#) (VkViewport value)  
*Set the viewport.*
- VkRect2D [get\\_scissor](#) () const  
*Get the scissor.*
- void [set\\_scissor](#) (VkRect2D value)  
*Set the scissor.*
- [sizing\\_mode](#) [get\\_sizing](#) () const  
*Get the sizing.*
- void [set\\_sizing](#) ([sizing\\_mode](#) value)  
*Set the sizing.*
- void [copy\\_to](#) ([render\\_pipeline](#) \*target) const  
*Copy pipeline configuration to target.*
- void [copy\\_from](#) (s\_ptr const &source)  
*Copy pipeline configuration from source.*
- void [set\\_line\\_width](#) (r32 value)  
*Set the line width.*
- r32 [get\\_line\\_width](#) () const  
*Get the line width.*
- bool [auto\\_line\\_width](#) () const  
*Check if auto line width is active.*
- void [set\\_auto\\_line\\_width](#) (bool value=true)  
*Set the auto line width.*
- void [set\\_line\\_width](#) (VkCommandBuffer cmd\_buf)  
*Set the line width.*

## Public Member Functions inherited from [lava::pipeline](#)

- [pipeline](#) (device::ptr device, VkPipelineCache pipeline\_cache=0)  
*Construct a new pipeline.*
- **~pipeline** () override  
*Destroy the pipeline.*
- bool [create](#) ()

- Create a new pipeline.*

  - void **destroy** ()

*Destroy the pipeline.*
- void **set\_active** (bool value=true)

*Set pipeline active.*
- bool **activated** () const

*Check if pipeline is active.*
- void **toggle** ()

*Toggle activation.*
- void **set\_auto\_bind** (bool value=true)

*Set auto bind.*
- bool **auto\_bind** () const

*Check if auto bind is enabled.*
- bool **ready** () const

*Check if pipeline is ready.*
- VkPipeline **get** () const

*Get the pipeline.*
- device::ptr **get\_device** ()

*Get the device.*
- pipeline\_layout::s\_ptr **get\_layout** () const

*Get the layout.*
- void **set\_layout** (pipeline\_layout::s\_ptr const &value)

*Set the layout.*

### Public Member Functions inherited from lava::entity

- **entity** ()
- Construct a new entity.*
- id::ref **get\_id** () const
- Get the id of entity.*

### Public Member Functions inherited from lava::no\_copy\_no\_move

- **no\_copy\_no\_move** ()=default
- Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete
- No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete
- No move.*

### Public Member Functions inherited from lava::interface

- virtual ~**interface** ()=default
- Destroy the interface.*

### Static Public Member Functions

- static s\_ptr **make** (device::ptr device, VkPipelineCache pipeline\_cache=0)
- Make a new render pipeline.*

**Public Attributes**

- [create\\_func](#) **on\_create**

*Called on render pipeline create.*

**Public Attributes inherited from [lava::pipeline](#)**

- [process\\_func](#) **on\_process**

*Called on pipeline process.*

**Additional Inherited Members****Protected Member Functions inherited from [lava::pipeline](#)****Protected Attributes inherited from [lava::pipeline](#)**

- [device::ptr](#) **m\_device** = nullptr  
*Vulkan device.*
- VkPipeline **m\_vk\_pipeline** = VK\_NULL\_HANDLE  
*Vulkan pipeline.*
- VkPipelineCache **m\_pipeline\_cache** = VK\_NULL\_HANDLE  
*Vulkan pipeline cache.*
- [pipeline\\_layout::s\\_ptr](#) **m\_layout**  
*Pipeline layout.*

**4.96.1 Detailed Description**

Render pipeline (Graphics)

**4.96.2 Constructor & Destructor Documentation****4.96.2.1 render\_pipeline()**

```
lava::render_pipeline::render_pipeline (
    device::ptr device,
    VkPipelineCache pipeline_cache) [explicit]
```

Construct a new render pipeline.

**Parameters**

<i>device</i>	Vulkan device
<i>pipeline_cache</i>	Pipeline cache

**4.96.3 Member Function Documentation****4.96.3.1 add()**

```
void lava::render_pipeline::add (
    shader_stage::s_ptr const & shader_stage) [inline]
```

Add shader stage.



## Parameters

<i>shader_stage</i>	Shader stage
---------------------	--------------

**4.96.3.2 add\_color\_blend\_attachment()**

```
void lava::render_pipeline::add_color_blend_attachment (
    VkPipelineColorBlendAttachmentState const & attachment)
```

Add color blend attachment.

## Parameters

<i>attachment</i>	Pipeline color blend attachment state
-------------------	---------------------------------------

**4.96.3.3 add\_dynamic\_state()**

```
void lava::render_pipeline::add_dynamic_state (
    VkDynamicState state)
```

Add a dynamic state.

## Parameters

<i>state</i>	Dynamic state
--------------	---------------

**4.96.3.4 add\_shader()**

```
bool lava::render_pipeline::add_shader (
    c_data::ref data,
    VkShaderStageFlagBits stage) [inline]
```

Add shader.

## Parameters

<i>data</i>	Shader data
<i>stage</i>	Shader stage flag bits

## Returns

Add was successful or failed

**4.96.3.5 add\_shader\_stage()**

```
bool lava::render_pipeline::add_shader_stage (
    c_data::ref data,
    VkShaderStageFlagBits stage)
```

Add shader stage.

**Parameters**

<i>data</i>	Shader data
<i>stage</i>	Shader stage flag bits

**Returns**

Add was successful or failed

**4.96.3.6 auto\_line\_width()**

```
bool lava::render_pipeline::auto_line_width () const [inline]
```

Check if auto line width is active.

**Returns**

Auto line width is enabled or not

**4.96.3.7 auto\_sizing()**

```
bool lava::render_pipeline::auto_sizing () const [inline]
```

Get the auto sizing state.

**Returns**

Auto sizing is enabled or not

**4.96.3.8 bind()**

```
void lava::render_pipeline::bind (  
    VkCommandBuffer cmd_buf) [override], [virtual]
```

Bind the pipeline.

**Parameters**

<i>cmd_buf</i>	Command buffer
----------------	----------------

Implements [lava::pipeline](#).

**4.96.3.9 copy\_from()**

```
void lava::render_pipeline::copy_from (  
    s\_ptr const & source) [inline]
```

Copy pipeline configuration from source.

## Parameters

<i>source</i>	Render pipeline
---------------	-----------------

**4.96.3.10 copy\_to()**

```
void lava::render_pipeline::copy_to (
    render_pipeline * target) const
```

Copy pipeline configuration to target.

## Parameters

<i>target</i>	Render pipeline
---------------	-----------------

**4.96.3.11 create()**

```
bool lava::render_pipeline::create (
    VkRenderPass pass) [inline]
```

Create a new render pipeline.

## Parameters

<i>pass</i>	Vulkan render pass
-------------	--------------------

## Returns

Create was successful or failed

**4.96.3.12 get\_line\_width()**

```
r32 lava::render_pipeline::get_line_width () const [inline]
```

Get the line width.

## Returns

r32 Line width

**4.96.3.13 get\_render\_pass()**

```
VkRenderPass lava::render_pipeline::get_render_pass () const [inline]
```

Get the render pass.

## Returns

VkRenderPass Render pass

#### 4.96.3.14 get\_scissor()

```
VkRect2D lava::render_pipeline::get_scissor () const [inline]
```

Get the scissor.

Returns

VkRect2D Scissor rectangle

#### 4.96.3.15 get\_shader\_stages()

```
shader_stage::s_list const & lava::render_pipeline::get_shader_stages () const [inline]
```

Get the shader stages.

Returns

shader\_stage::s\_list const& List of shader stages

#### 4.96.3.16 get\_sizing()

```
sizing_mode lava::render_pipeline::get_sizing () const [inline]
```

Get the sizing.

Returns

sizing\_mode Sizing mode

#### 4.96.3.17 get\_subpass()

```
index lava::render_pipeline::get_subpass () const [inline]
```

Get the subpass.

Returns

index Index of subpass

#### 4.96.3.18 get\_viewport()

```
VkViewport lava::render_pipeline::get_viewport () const [inline]
```

Get the viewport.

Returns

VkViewport Vulkan viewport

#### 4.96.3.19 make()

```
static s_ptr lava::render_pipeline::make (  
    device::ptr device,  
    VkPipelineCache pipeline_cache = 0) [inline], [static]
```

Make a new render pipeline.

## Parameters

<i>device</i>	Vulkan device
<i>pipeline_cache</i>	Pipeline cache

## Returns

s\_ptr Shared pointer to render pipeline

**4.96.3.20 set()**

```
void lava::render_pipeline::set (  
    VkRenderPass pass) [inline]
```

## See also

[set\\_render\\_pass](#)

**4.96.3.21 set\_auto\_line\_width()**

```
void lava::render_pipeline::set_auto_line_width (  
    bool value = true) [inline]
```

Set the auto line width.

## Parameters

<i>value</i>	Enable state
--------------	--------------

**4.96.3.22 set\_auto\_size()**

```
void lava::render_pipeline::set_auto_size (  
    bool value = true) [inline]
```

Set the auto size.

## Parameters

<i>value</i>	Enable state
--------------	--------------

**4.96.3.23 set\_depth\_compare\_op()**

```
void lava::render_pipeline::set_depth_compare_op (  
    VkCompareOp compare_op)
```

Set the depth compare operation.

## Parameters

<i>compare_op</i>	Depth compare operation
-------------------	-------------------------

**4.96.3.24 set\_depth\_test\_and\_write()**

```
void lava::render_pipeline::set_depth_test_and_write (
    bool test_enable = true,
    bool write_enable = true)
```

Set the depth test and write.

## Parameters

<i>test_enable</i>	Enable depth test
<i>write_enable</i>	Enable depth write

**4.96.3.25 set\_dynamic\_states()**

```
void lava::render_pipeline::set_dynamic_states (
    VkDynamicStates const & states)
```

Set the dynamic states.

## Parameters

<i>states</i>	List of dynamic states
---------------	------------------------

**4.96.3.26 set\_input\_topology()**

```
void lava::render_pipeline::set_input_topology (
    VkPrimitiveTopology const & topology)
```

Set the input assembler's topology.

## Parameters

<i>topology</i>	Enum describing polygon primitives
-----------------	------------------------------------

**4.96.3.27 set\_line\_width()** [1/2]

```
void lava::render_pipeline::set_line_width (
    r32 value) [inline]
```

Set the line width.

## Parameters

<i>value</i>	Line width
--------------	------------

**4.96.3.28 set\_line\_width()** [2/2]

```
void lava::render_pipeline::set_line_width (
    VkCommandBuffer cmd_buf) [inline]
```

Set the line width.

## Parameters

<i>cmd_buf</i>	Command buffer
----------------	----------------

**4.96.3.29 set\_rasterization\_cull\_mode()**

```
void lava::render_pipeline::set_rasterization_cull_mode (
    VkCullModeFlags cull_mode)
```

Set the rasterization cull mode.

## Parameters

<i>cull_mode</i>	Cull mode flags
------------------	-----------------

**4.96.3.30 set\_rasterization\_front\_face()**

```
void lava::render_pipeline::set_rasterization_front_face (
    VkFrontFace front_face)
```

Set the rasterization front face.

## Parameters

<i>front_face</i>	Front face
-------------------	------------

**4.96.3.31 set\_rasterization\_polygon\_mode()**

```
void lava::render_pipeline::set_rasterization_polygon_mode (
    VkPolygonMode polygon_mode)
```

Set the rasterization polygon mode.

## Parameters

<i>polygon_mode</i>	Polygon mode
---------------------	--------------

**4.96.3.32 set\_render\_pass()**

```
void lava::render_pipeline::set_render_pass (
    VkRenderPass pass) [inline]
```

Set the render pass.

## Parameters

<i>pass</i>	Render pass
-------------	-------------

**4.96.3.33 set\_scissor()**

```
void lava::render_pipeline::set_scissor (
    VkRect2D value) [inline]
```

Set the scissor.

## Parameters

<i>value</i>	Scissor rectangle
--------------	-------------------

**4.96.3.34 set\_sizing()**

```
void lava::render_pipeline::set_sizing (
    sizing_mode value) [inline]
```

Set the sizing.

## Parameters

<i>value</i>	Sizing mode
--------------	-------------

**4.96.3.35 set\_subpass()**

```
void lava::render_pipeline::set_subpass (
    index value) [inline]
```

Set the subpass.



## Parameters

<i>value</i>	Index of subpass
--------------	------------------

**4.96.3.36 set\_vertex\_input\_attribute()**

```
void lava::render_pipeline::set_vertex_input_attribute (
    VkVertexInputAttributeDescription const & attribute)
```

Set the vertex input attribute.

## Parameters

<i>attribute</i>	Vertex input attribute description
------------------	------------------------------------

**4.96.3.37 set\_vertex\_input\_attributes()**

```
void lava::render_pipeline::set_vertex_input_attributes (
    VkVertexInputAttributeDescriptions const & attributes)
```

Set the vertex input attributes.

## Parameters

<i>attributes</i>	List of vertex input attributes descriptions
-------------------	--

**4.96.3.38 set\_vertex\_input\_binding()**

```
void lava::render_pipeline::set_vertex_input_binding (
    VkVertexInputBindingDescription const & description)
```

Set the vertex input binding.

## Parameters

<i>description</i>	Vertex input binding description
--------------------	----------------------------------

**4.96.3.39 set\_vertex\_input\_bindings()**

```
void lava::render_pipeline::set_vertex_input_bindings (
    VkVertexInputBindingDescriptions const & descriptions)
```

Set the vertex input bindings.

## Parameters

<i>descriptions</i>	List of vertex input binding descriptions
---------------------	---

**4.96.3.40 set\_viewport()**

```
void lava::render_pipeline::set_viewport (
    VkViewport value) [inline]
```

Set the viewport.

## Parameters

<i>value</i>	Vulkan viewport
--------------	-----------------

**4.96.3.41 set\_viewport\_and\_scissor()**

```
void lava::render_pipeline::set_viewport_and_scissor (
    VkCommandBuffer cmd_buf,
    uv2 size)
```

Set the viewport and scissor.

## Parameters

<i>cmd_buf</i>	Command buffer
<i>size</i>	Viewport and scissor size

The documentation for this struct was generated from the following file:

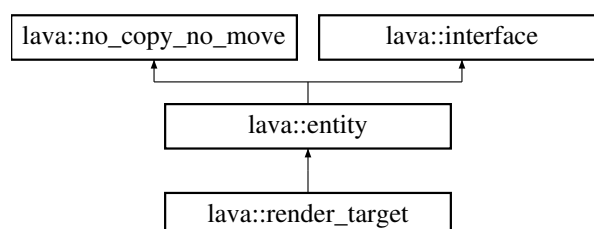
- liblava/block/[render\\_pipeline.hpp](#)

**4.97 lava::render\_target Struct Reference**

Render target.

```
#include <render_target.hpp>
```

Inheritance diagram for lava::render\_target:



## Public Types

- using **s\_ptr** = std::shared\_ptr<render\_target>  
*Shared pointer to render target.*
- using **swapchain\_start\_func** = std::function<bool()>  
*Swapchain start function.*
- using **swapchain\_stop\_func** = std::function<void()>  
*Swapchain stop function.*
- using **create\_attachments\_func** = std::function<VkAttachments()>  
*Create attachments function.*
- using **destroy\_attachments\_func** = std::function<void()>  
*Destroy attachments function.*

## Public Member Functions

- bool **create** (device::ptr device, VkSurfaceKHR surface, VkSurfaceFormatKHR format, uv2 size, bool v\_↵ sync=false, bool triple\_buffer=true)  
*Create a new render target.*
- void **destroy** ()  
*Destroy the render target.*
- uv2 **get\_size** () const  
*Get the size of the render target.*
- bool **resize** (uv2 new\_size)  
*Resize the render target.*
- ui32 **get\_frame\_count** () const  
*Get the frame count.*
- bool **reload\_request** () const  
*Check if render target requests a reload.*
- void **reload** ()  
*Reload the render target.*
- device::ptr **get\_device** ()  
*Get the device.*
- swapchain \* **get\_swapchain** ()  
*Get the swapchain.*
- VkFormat **get\_format** () const  
*Get the format.*
- image::s\_list const & **get\_backbuffers** () const  
*Get the backbuffers.*
- image::s\_ptr **get\_backbuffer** (index index)  
*Get the backbuffer by frame index.*
- VkImage **get\_backbuffer\_image** (index index)  
*Get the backbuffer image by index.*
- VkImage **get\_image** (index index)
- void **add\_callback** (target\_callback::c\_ptr callback)  
*Add callback.*
- void **remove\_callback** (target\_callback::c\_ptr callback)  
*Remove callback.*

## Public Member Functions inherited from [lava::entity](#)

- **entity** ()  
*Construct a new entity.*
- **id::ref get\_id** () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static **s\_ptr make** ()  
*Make a new render target.*

## Public Attributes

- **swapchain\_start\_func on\_swapchain\_start**  
*Called on swapchain start.*
- **swapchain\_stop\_func on\_swapchain\_stop**  
*Called on swapchain stop.*
- **create\_attachments\_func on\_create\_attachments**  
*Called on create attachments.*
- **destroy\_attachments\_func on\_destroy\_attachments**  
*Called on destroy attachments.*

### 4.97.1 Detailed Description

Render target.

### 4.97.2 Member Function Documentation

#### 4.97.2.1 add\_callback()

```
void lava::render_target::add_callback (
    target\_callback::c\_ptr callback) [inline]
```

Add callback.

## Parameters

<i>callback</i>	Target callback
-----------------	-----------------

## 4.97.2.2 create()

```
bool lava::render_target::create (
    device::ptr device,
    VkSurfaceKHR surface,
    VkSurfaceFormatKHR format,
    uv2 size,
    bool v_sync = false,
    bool triple_buffer = true)
```

Create a new render target.

## Parameters

<i>device</i>	Vulkan device
<i>surface</i>	Vulkan surface
<i>format</i>	Surface format
<i>size</i>	Size of target
<i>v_sync</i>	V-Sync enabled
<i>triple_buffer</i>	VK_PRESENT_MODE_MAILBOX_KHR preferred over VK_PRESENT_MODE_IMMEDIATE_KHR

## Returns

Create was successful or failed

## 4.97.2.3 get\_backbuffer()

```
image::s_ptr lava::render_target::get_backbuffer (
    index index) [inline]
```

Get the backbuffer by frame index.

## Parameters

<i>index</i>	Frame index
--------------	-------------

## Returns

[image::s\\_ptr](#) Backbuffer image

## 4.97.2.4 get\_backbuffer\_image()

```
VkImage lava::render_target::get_backbuffer_image (
    index index) [inline]
```

Get the backbuffer image by index.

## Parameters

<i>index</i>	Frame index
--------------	-------------

## Returns

VkImage Vulkan image

**4.97.2.5 get\_backbuffers()**

```
image::s_list const & lava::render_target::get_backbuffers () const [inline]
```

Get the backbuffers.

## Returns

[image::s\\_list](#) const& List of backbuffer images

**4.97.2.6 get\_device()**

```
device::ptr lava::render_target::get_device () [inline]
```

Get the device.

## Returns

[device::ptr](#) Vulkan device

**4.97.2.7 get\_format()**

```
VkFormat lava::render_target::get_format () const [inline]
```

Get the format.

## Returns

VkFormat Target format

**4.97.2.8 get\_frame\_count()**

```
ui32 lava::render_target::get_frame_count () const [inline]
```

Get the frame count.

## Returns

ui32 Number of frames

#### 4.97.2.9 get\_image()

```
VkImage lava::render_target::get_image (
    index index) [inline]
```

See also

[get\\_backbuffer\\_image](#)

#### 4.97.2.10 get\_size()

```
uv2 lava::render_target::get_size () const [inline]
```

Get the size of the render target.

Returns

uv2 Size of render target

#### 4.97.2.11 get\_swapchain()

```
swapchain * lava::render_target::get_swapchain () [inline]
```

Get the swapchain.

Returns

swapchain\* Target swapchain

#### 4.97.2.12 make()

```
static s_ptr lava::render_target::make () [inline], [static]
```

Make a new render target.

Returns

s\_ptr Shared pointer to render target

#### 4.97.2.13 reload\_request()

```
bool lava::render_target::reload_request () const [inline]
```

Check if render target requests a reload.

Returns

Request reload or not

#### 4.97.2.14 remove\_callback()

```
void lava::render_target::remove_callback (
    target_callback::c_ptr callback) [inline]
```

Remove callback.

## Parameters

<i>callback</i>	Target callback
-----------------	-----------------

4.97.2.15 `resize()`

```
bool lava::render_target::resize (
    uv2 new_size) [inline]
```

Resize the render target.

## Parameters

<i>new_size</i>	New render target size
-----------------	------------------------

## Returns

Resize was successful or failed

The documentation for this struct was generated from the following file:

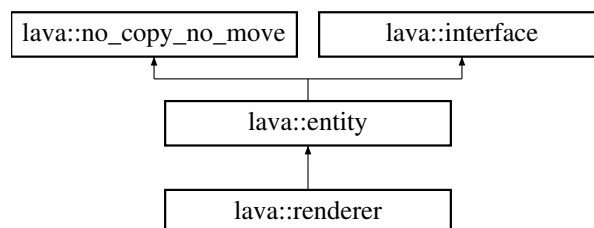
- [liblava/frame/render\\_target.hpp](#)

4.98 `lava::renderer` Struct Reference

Plain renderer.

```
#include <renderer.hpp>
```

Inheritance diagram for `lava::renderer`:



## Public Types

- using **ptr** = [renderer\\*](#)  
*Pointer to renderer.*
- using **destroy\_func** = `std::function<void()>`  
*Destroy function.*



## Public Member Functions

- bool [create](#) ([swapchain](#) \*target)  
*Create a new renderer.*
- void **destroy** ()  
*Destroy the renderer.*
- [optional\\_index begin\\_frame](#) ()  
*Begin to render a frame.*
- bool [end\\_frame](#) ([VkCommandBuffers](#) const &cmd\_buffers)  
*End of frame rendering.*
- bool [frame](#) ([VkCommandBuffers](#) const &cmd\_buffers)  
*Render a frame.*
- [index get\\_frame](#) () const  
*Get the current frame index.*
- [device::ptr get\\_device](#) ()  
*Get the device.*

## Public Member Functions inherited from [lava::entity](#)

- **entity** ()  
*Construct a new entity.*
- [id::ref get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Public Attributes

- [VkSemaphores](#) **user\_frame\_wait\_semaphores**  
*The frame waits additionally for these semaphores (Usefully for additional CommandBuffers)*
- [VkPipelineStageFlagsList](#) **user\_frame\_wait\_stages**  
*To user\_frame\_wait\_semaphores corresponding pipeline wait stages.*
- [VkSemaphores](#) **user\_frame\_signal\_semaphores**  
*The frame additionally signals these semaphores (Usefully for additional CommandBuffers)*
- **destroy\_func on\_destroy**  
*Called on renderer destroy.*
- bool **active** = true  
*Active state.*

### 4.98.1 Detailed Description

Plain renderer.

### 4.98.2 Member Function Documentation

#### 4.98.2.1 begin\_frame()

```
optional_index lava::renderer::begin_frame ()
```

Begin to render a frame.

##### Returns

optional\_index Frame index

#### 4.98.2.2 create()

```
bool lava::renderer::create (
    swapchain * target)
```

Create a new renderer.

##### Parameters

<i>target</i>	Swapchain target
---------------	------------------

##### Returns

Create was successful or failed

#### 4.98.2.3 end\_frame()

```
bool lava::renderer::end_frame (
    VkCommandBuffers const & cmd_buffers)
```

End of frame rendering.

##### Parameters

<i>cmd_buffers</i>	List of command buffers
--------------------	-------------------------

##### Returns

End was successful or failed

#### 4.98.2.4 frame()

```
bool lava::renderer::frame (
    VkCommandBuffers const & cmd_buffers) [inline]
```

Render a frame.

## Parameters

<code>cmd_buffers</code>	List of command buffers
--------------------------	-------------------------

## Returns

Render was successful or failed

#### 4.98.2.5 get\_device()

```
device::ptr lava::renderer::get_device () [inline]
```

Get the device.

## Returns

`device::ptr` Vulkan device

#### 4.98.2.6 get\_frame()

```
index lava::renderer::get_frame () const [inline]
```

Get the current frame index.

## Returns

index Frame index

The documentation for this struct was generated from the following file:

- [liblava/frame/renderer.hpp](#)

## 4.99 lava::driver::result Struct Reference

Driver result.

```
#include <driver.hpp>
```

### Public Attributes

- `i32 driver = 0`  
*Run result.*
- `i32 selected = 0`  
*Selected stage.*

### 4.99.1 Detailed Description

Driver result.

The documentation for this struct was generated from the following file:

- [liblava/frame/driver.hpp](#)

## 4.100 `lava::reversion_wrapper< T >` Struct Template Reference

Reversion Wrapper.

```
#include <misc.hpp>
```

### Public Attributes

- **T & iterable**  
*Iterable to wrap.*

### 4.100.1 Detailed Description

```
template<typename T>
struct lava::reversion_wrapper< T >
```

Reversion Wrapper.

#### Template Parameters

<i>T</i>	Type to iterate
----------	-----------------

The documentation for this struct was generated from the following file:

- [liblava/core/misc.hpp](#)

## 4.101 `lava::run_time` Struct Reference

Run time.

```
#include <time.hpp>
```

### Public Attributes

- `ms current {0}`  
*Current milliseconds.*
- `ms clock {16}`  
*Clock milliseconds.*
- `ms system {0}`  
*System milliseconds.*
- `ms delta {0}`  
*Delta milliseconds.*
- `ms fix_delta {0}`  
*Fix delta milliseconds (0 = deactivated)*
- `r32 speed = 1.f`  
*Speed factor.*
- `bool paused = false`  
*Paused run time.*

#### 4.101.1 Detailed Description

Run time.

The documentation for this struct was generated from the following file:

- [liblava/core/time.hpp](#)

## 4.102 `lava::scoped_label< T >` Struct Template Reference

Scoped debug util label.

```
#include <debug_utils.hpp>
```

### Public Member Functions

- `scoped_label` (T scope, `name` label, `v4` color=`v4(0.f)`)  
*Construct a new scoped label.*
- `~scoped_label` ()  
*Destroy the scoped label.*

#### 4.102.1 Detailed Description

```
template<typename T>
struct lava::scoped_label< T >
```

Scoped debug util label.

## Template Parameters

<i>T</i>	VkCommandBuffer or VkQueue
----------	----------------------------

## 4.102.2 Constructor & Destructor Documentation

### 4.102.2.1 scoped\_label()

```
template<typename T >
lava::scoped_label< T >::scoped_label (
    T scope,
    name label,
    v4 color = v4(0.f)) [inline]
```

Construct a new scoped label.

## Parameters

<i>scope</i>	Scoped label
<i>label</i>	Name of label
<i>color</i>	Color of label

The documentation for this struct was generated from the following file:

- liblava/base/[debug\\_utils.hpp](#)

## 4.103 lava::scroll\_event Struct Reference

Scroll event.

```
#include <input.hpp>
```

## Public Types

- using **ref** = [scroll\\_event](#) const&  
*Reference to scroll event.*
- using **func** = std::function<bool([ref](#))>  
*Scroll event function.*
- using **listeners** = std::map<[id](#), [func](#)>  
*List of scroll event listeners.*
- using **list** = std::vector<[scroll\\_event](#)>  
*List of scroll events.*

### Public Attributes

- [id](#) **sender**  
*Sender id.*
- [scroll\\_offset](#) **offset**  
*Input scroll offset.*

### 4.103.1 Detailed Description

Scroll event.

The documentation for this struct was generated from the following file:

- [liblava/frame/input.hpp](#)

## 4.104 lava::scroll\_offset Struct Reference

Input scroll offset.

```
#include <input.hpp>
```

### Public Attributes

- [r64](#) **x** = 0.0  
*X offset.*
- [r64](#) **y** = 0.0  
*Y offset.*

### 4.104.1 Detailed Description

Input scroll offset.

The documentation for this struct was generated from the following file:

- [liblava/frame/input.hpp](#)

## 4.105 lava::semantic\_version Struct Reference

Semantic version.

```
#include <version.hpp>
```

### Public Member Functions

- auto **operator**<=> ([semantic\\_version](#) const &) const =default  
*Default compare operators.*

## Public Attributes

- **ui32 major** = LAVA\_VERSION\_MAJOR  
*Major version.*
- **ui32 minor** = LAVA\_VERSION\_MINOR  
*Minor version.*
- **ui32 patch** = LAVA\_VERSION\_PATCH  
*Patch version.*

### 4.105.1 Detailed Description

Semantic version.

The documentation for this struct was generated from the following file:

- liblava/core/[version.hpp](#)

## 4.106 lava::pipeline::shader\_stage Struct Reference

Shader stage.

```
#include <pipeline.hpp>
```

## Public Types

- using **s\_ptr** = std::shared\_ptr<[shader\\_stage](#)>  
*Shared pointer to shader stage.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of shader stages.*

## Public Member Functions

- **shader\_stage** ()  
*Construct a new shader stage.*
- **~shader\_stage** ()  
*Destroy the shader stage.*
- void **set\_stage** (VkShaderStageFlagBits [stage](#))  
*Set the stage.*
- void **add\_specialization\_entry** (VkSpecializationMapEntry const &specialization)  
*Add specialization entry.*
- bool **create** ([device::ptr](#) device, [c\\_data::ref](#) shader\_data, [c\\_data::ref](#) specialization\_data=[data](#)())  
*Create a new shader stage.*
- void **destroy** ()  
*Destroy the shader stage.*
- VkPipelineShaderStageCreateInfo const & **get\_create\_info** () const  
*Get the create info.*



## Static Public Member Functions

- static [s\\_ptr make](#) (VkShaderStageFlagBits [stage](#))  
*Make a new pipeline shader stage.*

### 4.106.1 Detailed Description

Shader stage.

### 4.106.2 Member Function Documentation

#### 4.106.2.1 add\_specialization\_entry()

```
void lava::pipeline::shader_stage::add_specialization_entry (
    VkSpecializationMapEntry const & specialization)
```

Add specialization entry.

##### Parameters

<i>specialization</i>	Specialization map entry
-----------------------	--------------------------

#### 4.106.2.2 create()

```
bool lava::pipeline::shader_stage::create (
    device::ptr device,
    c_data::ref shader_data,
    c_data::ref specialization_data = data())
```

Create a new shader stage.

##### Parameters

<i>device</i>	Vulkan device
<i>shader_data</i>	Shader data
<i>specialization_data</i>	Specialization data

##### Returns

Create was successful or failed

#### 4.106.2.3 get\_create\_info()

```
VkPipelineShaderStageCreateInfo const & lava::pipeline::shader_stage::get_create_info () const
[inline]
```

Get the create info.

##### Returns

VkPipelineShaderStageCreateInfo const& Pipeline shader stage create information

#### 4.106.2.4 make()

```
static s_ptr lava::pipeline::shader_stage::make (  
    VkShaderStageFlagBits stage) [inline], [static]
```

Make a new pipeline shader stage.

## Parameters

<i>stage</i>	Shader stage flag bits
--------------	------------------------

## Returns

s\_ptr Shared pointer to shader stage

## 4.106.2.5 set\_stage()

```
void lava::pipeline::shader_stage::set_stage (
    VkShaderStageFlagBits stage) [inline]
```

Set the stage.

## Parameters

<i>stage</i>	Shader stage flag bits
--------------	------------------------

The documentation for this struct was generated from the following file:

- liblava/block/[pipeline.hpp](#)

## 4.107 lava::stage Struct Reference

Stage.

```
#include <driver.hpp>
```

## Public Types

- using **map** = std::map<[index](#), [stage](#)\*>  
*Map of stages.*
- using **func** = std::function<[i32](#)(argh::parser)>  
*Stage function.*

## Public Member Functions

- [stage](#) ([ui32](#) id, [string\\_ref](#) name, [func](#) func)  
*Construct a new stage.*

## Public Attributes

- [index](#) **id** = 0  
*Stage id.*
- [string](#) **name**  
*Stage name.*
- [func](#) **on\_func**  
*Called on stage run.*

### 4.107.1 Detailed Description

Stage.

### 4.107.2 Constructor & Destructor Documentation

#### 4.107.2.1 stage()

```
lava::stage::stage (
    ui32 id,
    string_ref name,
    func func) [explicit]
```

Construct a new stage.

#### Parameters

<i>id</i>	Stage id
<i>name</i>	Stage name
<i>func</i>	Stage function

The documentation for this struct was generated from the following file:

- liblava/frame/[driver.hpp](#)

## 4.108 lava::staging Struct Reference

Texture staging.

```
#include <texture.hpp>
```

#### Public Types

- using **ptr** = [staging\\*](#)  
*Pointer to staging.*

#### Public Member Functions

- void [add](#) ([texture::s\\_ptr](#) texture)  
*Add texture for staging.*
- bool [stage](#) (VkCommandBuffer cmd\_buf, [index](#) frame)  
*Stage textures.*
- void **clear** ()  
*Clear staging.*
- bool [busy](#) () const  
*Check if staging is busy.*

## 4.108.1 Detailed Description

Texture staging.

## 4.108.2 Member Function Documentation

### 4.108.2.1 add()

```
void lava::staging::add (  
    texture::s_ptr texture) [inline]
```

Add texture for staging.

#### Parameters

<i>texture</i>	Texture to stage
----------------	------------------

### 4.108.2.2 busy()

```
bool lava::staging::busy () const [inline]
```

Check if staging is busy.

#### Returns

Staging is busy or not

### 4.108.2.3 stage()

```
bool lava::staging::stage (  
    VkCommandBuffer cmd_buf,  
    index frame)
```

Stage textures.

#### Parameters

<i>cmd_buf</i>	Command buffer
<i>frame</i>	Frame index

#### Returns

Stage was successful or failed

The documentation for this struct was generated from the following file:

- [liblava/resource/texture.hpp](#)

## 4.109 `lava::window::state` Struct Reference

Window state.

```
#include <window.hpp>
```

### Public Types

- using **ref** = `state` const&  
*Reference to window state.*
- using **optional** = std::optional<`window::state`>  
*Optional window state.*

### Public Member Functions

- **state** ()  
*Construct a new state.*

### Public Attributes

- `i32` **x** = 0  
*Window X position.*
- `i32` **y** = 0  
*Window Y position.*
- `ui32` **width** = 0  
*Window width.*
- `ui32` **height** = 0  
*Window height.*
- bool **fullscreen** = false  
*Fullscreen window.*
- bool **floating** = false  
*Floating window.*
- bool **resizable** = true  
*Resizable window.*
- bool **decorated** = true  
*Decorated window.*
- bool **maximized** = false  
*Maximized window.*
- `index` **monitor** = 0  
*Monitor of window.*

### 4.109.1 Detailed Description

Window state.

The documentation for this struct was generated from the following file:

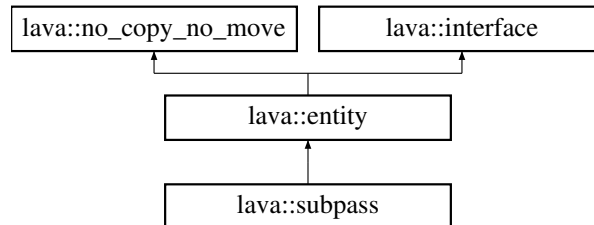
- liblava/frame/[window.hpp](#)

## 4.110 lava::subpass Struct Reference

Subpass.

```
#include <subpass.hpp>
```

Inheritance diagram for lava::subpass:



### Public Types

- using **s\_ptr** = std::shared\_ptr<subpass>  
*Shared pointer to subpass.*
- using **s\_list** = std::vector<s\_ptr>  
*List of subpasses.*

### Public Member Functions

- **subpass** ()  
*Construct a new subpass.*
- void **destroy** ()  
*Destroy the subpass.*
- void **add** (render\_pipeline::s\_ptr const &pipeline)  
*Add a render pipeline to the back of the subpass.*
- void **add\_front** (render\_pipeline::s\_ptr const &pipeline)  
*Add a render pipeline to the front of the subpass.*
- void **remove** (render\_pipeline::s\_ptr pipeline)  
*Remove the render pipeline.*
- void **clear\_pipelines** ()  
*Clear all pipelines.*
- void **process** (VkCommandBuffer cmd\_buf, uv2 size)  
*Process the subpass.*
- VkSubpassDescription const & **get\_description** () const  
*Get the description.*
- void **set** (VkPipelineBindPoint pipeline\_bind\_point)  
*Set pipeline bind point.*
- void **set\_color\_attachment** (index attachment, VkImageLayout layout)  
*Set the color attachment.*
- void **set\_color\_attachment** (VkAttachmentReference attachment)  
*Set the color attachment.*
- void **set\_color\_attachments** (VkAttachmentReferences const &attachments)  
*Set the color attachments.*
- void **set\_depth\_stencil\_attachment** (index attachment, VkImageLayout layout)

- Set the depth stencil attachment.*
  - void [set\\_depth\\_stencil\\_attachment](#) (VkAttachmentReference [attachment](#))
- Set the depth stencil attachment.*
  - void [set\\_input\\_attachment](#) (index [attachment](#), VkImageLayout layout)
- Set the input attachment.*
  - void [set\\_input\\_attachment](#) (VkAttachmentReference [attachment](#))
- Set the input attachment.*
  - void [set\\_input\\_attachments](#) (VkAttachmentReferences const &attachments)
- Set the input attachments.*
  - void [set\\_resolve\\_attachment](#) (index [attachment](#), VkImageLayout layout)
- Set the resolve attachment.*
  - void [set\\_resolve\\_attachment](#) (VkAttachmentReference [attachment](#))
- Set the resolve attachment.*
  - void [set\\_resolve\\_attachments](#) (VkAttachmentReferences const &attachments)
- Set the resolve attachments.*
  - void [add\\_preserve\\_attachment](#) (index [attachment](#))
- Add preserve attachment.*
  - void [set\\_preserve\\_attachments](#) (index\_list const &attachments)
- Set the preserve attachments.*
  - void [set\\_active](#) (bool value=true)
- Activate or deactivate the subpass.*
  - bool [activated](#) () const
- Check if subpass is active.*

## Public Member Functions inherited from [lava::entity](#)

- [entity](#) ()
- Construct a new entity.*
- [id::ref get\\_id](#) () const
- Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default
- Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete
- No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete
- No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual [~interface](#) ()=default
- Destroy the interface.*

## Static Public Member Functions

- static [s\\_ptr make](#) (VkPipelineBindPoint pipeline\_bind\_point=VK\_PIPELINE\_BIND\_POINT\_GRAPHICS)
- Make a new subpass.*



### 4.110.1 Detailed Description

Subpass.

### 4.110.2 Member Function Documentation

#### 4.110.2.1 activated()

```
bool lava::subpass::activated () const [inline]
```

Check if subpass is active.

##### Returns

Subpass is active or not

#### 4.110.2.2 add()

```
void lava::subpass::add (  
    render_pipeline::s_ptr const & pipeline) [inline]
```

Add a render pipeline to the back of the subpass.

##### Parameters

<i>pipeline</i>	Render pipeline
-----------------	-----------------

#### 4.110.2.3 add\_front()

```
void lava::subpass::add_front (  
    render_pipeline::s_ptr const & pipeline) [inline]
```

Add a render pipeline to the frons of the subpass.

##### Parameters

<i>pipeline</i>	Render pipeline
-----------------	-----------------

#### 4.110.2.4 add\_preserve\_attachment()

```
void lava::subpass::add_preserve_attachment (  
    index attachment)
```

Add preserve attachment.

## Parameters

<i>attachment</i>	Index of attachment
-------------------	---------------------

**4.110.2.5 get\_description()**

```
VkSubpassDescription const & lava::subpass::get_description () const [inline]
```

Get the description.

## Returns

VkSubpassDescription const& Subpass description

**4.110.2.6 make()**

```
static s_ptr lava::subpass::make (
    VkPipelineBindPoint pipeline_bind_point = VK_PIPELINE_BIND_POINT_GRAPHICS) [inline],
[static]
```

Make a new subpass.

## Parameters

<i>pipeline_bind_point</i>	Pipeline bind point
----------------------------	---------------------

## Returns

s\_ptr Shared pointer to subpass

**4.110.2.7 process()**

```
void lava::subpass::process (
    VkCommandBuffer cmd_buf,
    uv2 size)
```

Process the subpass.

## Parameters

<i>cmd_buf</i>	Command buffer
<i>size</i>	Size of render pass

**4.110.2.8 remove()**

```
void lava::subpass::remove (
    render_pipeline::s_ptr pipeline)
```

Remove the render pipeline.

## Parameters

<i>pipeline</i>	Render pipeline
-----------------	-----------------

**4.110.2.9 set()**

```
void lava::subpass::set (
    VkPipelineBindPoint pipeline_bind_point) [inline]
```

Set pipeline bind point.

## Parameters

<i>pipeline_bind_point</i>	Pipeline bind point
----------------------------	---------------------

**4.110.2.10 set\_active()**

```
void lava::subpass::set_active (
    bool value = true) [inline]
```

Activate or deactivate the subpass.

## Parameters

<i>value</i>	Enable state
--------------	--------------

**4.110.2.11 set\_color\_attachment() [1/2]**

```
void lava::subpass::set_color_attachment (
    index attachment,
    VkImageLayout layout)
```

Set the color attachment.

## Parameters

<i>attachment</i>	Index of attachment
<i>layout</i>	Image layout

**4.110.2.12 set\_color\_attachment() [2/2]**

```
void lava::subpass::set_color_attachment (
    VkAttachmentReference attachment)
```

Set the color attachment.

## Parameters

<i>attachment</i>	Attachment reference
-------------------	----------------------

**4.110.2.13 set\_color\_attachments()**

```
void lava::subpass::set_color_attachments (
    VkAttachmentReferences const & attachments)
```

Set the color attachments.

## Parameters

<i>attachments</i>	List of attachment references
--------------------	-------------------------------

**4.110.2.14 set\_depth\_stencil\_attachment() [1/2]**

```
void lava::subpass::set_depth_stencil_attachment (
    index attachment,
    VkImageLayout layout)
```

Set the depth stencil attachment.

## Parameters

<i>attachment</i>	Index of attachment
<i>layout</i>	Image layout

**4.110.2.15 set\_depth\_stencil\_attachment() [2/2]**

```
void lava::subpass::set_depth_stencil_attachment (
    VkAttachmentReference attachment)
```

Set the depth stencil attachment.

## Parameters

<i>attachment</i>	Attachment reference
-------------------	----------------------

**4.110.2.16 set\_input\_attachment() [1/2]**

```
void lava::subpass::set_input_attachment (
    index attachment,
    VkImageLayout layout)
```

Set the input attachment.

## Parameters

<i>attachment</i>	Index of attachment
<i>layout</i>	Image layout

**4.110.2.17 set\_input\_attachment()** [2/2]

```
void lava::subpass::set_input_attachment (
    VkAttachmentReference attachment)
```

Set the input attachment.

## Parameters

<i>attachment</i>	Attachment reference
-------------------	----------------------

**4.110.2.18 set\_input\_attachments()**

```
void lava::subpass::set_input_attachments (
    VkAttachmentReferences const & attachments)
```

Set the input attachments.

## Parameters

<i>attachments</i>	List of attachment references
--------------------	-------------------------------

**4.110.2.19 set\_preserve\_attachments()**

```
void lava::subpass::set_preserve_attachments (
    index_list const & attachments)
```

Set the preserve attachments.

## Parameters

<i>attachments</i>	List of indices
--------------------	-----------------

**4.110.2.20 set\_resolve\_attachment()** [1/2]

```
void lava::subpass::set_resolve_attachment (
    index attachment,
    VkImageLayout layout)
```

Set the resolve attachment.

## Parameters

<i>attachment</i>	Index of attachment
<i>layout</i>	Image layout

**4.110.2.21 set\_resolve\_attachment() [2/2]**

```
void lava::subpass::set_resolve_attachment (
    VkAttachmentReference attachment)
```

Set the resolve attachment.

## Parameters

<i>attachment</i>	Attachment reference
-------------------	----------------------

**4.110.2.22 set\_resolve\_attachments()**

```
void lava::subpass::set_resolve_attachments (
    VkAttachmentReferences const & attachments)
```

Set the resolve attachments.

## Parameters

<i>attachments</i>	List of attachment references
--------------------	-------------------------------

The documentation for this struct was generated from the following file:

- [liblava/block/subpass.hpp](#)

**4.111 lava::subpass\_dependency Struct Reference**

Subpass dependency.

```
#include <subpass.hpp>
```

**Public Types**

- using **s\_ptr** = std::shared\_ptr<[subpass\\_dependency](#)>  
*Shared pointer to subpass dependency.*
- using **s\_list** = std::vector<[s\\_ptr](#)>  
*List of subpass dependencies.*

## Public Member Functions

- **subpass\_dependency ()**  
*Construct a new subpass dependency.*
- **VkSubpassDependency const & get\_dependency () const**  
*Get the dependency.*
- **void set\_subpass (ui32 src, ui32 dst)**  
*Set the subpass.*
- **void set\_src\_subpass (ui32 src)**  
*Set the source subpass.*
- **void set\_dst\_subpass (ui32 dst)**  
*Set the dst subpass.*
- **void set\_stage\_mask (VkPipelineStageFlags src, VkPipelineStageFlags dst)**  
*Set the stage mask.*
- **void set\_src\_stage\_mask (VkPipelineStageFlags mask)**  
*Set the source stage mask.*
- **void set\_dst\_stage\_mask (VkPipelineStageFlags mask)**  
*Set the destination stage mask.*
- **void set\_access\_mask (VkAccessFlags src, VkAccessFlags dst)**  
*Set the access mask.*
- **void set\_src\_access\_mask (VkAccessFlags mask)**  
*Set the src access mask.*
- **void set\_dst\_access\_mask (VkAccessFlags mask)**  
*Set the dst access mask.*
- **void set\_dependency\_flags (VkDependencyFlags flags)**  
*Set the dependency flags.*

## Static Public Member Functions

- **static s\_ptr make (ui32 src\_subpass, ui32 dst\_subpass, VkDependencyFlags dependency\_flags=VK\_DEPENDENCY\_BY\_REGION\_BIT)**  
*Make a new subpass dependency.*

### 4.111.1 Detailed Description

Subpass dependency.

### 4.111.2 Member Function Documentation

#### 4.111.2.1 get\_dependency()

```
VkSubpassDependency const & lava::subpass_dependency::get_dependency () const [inline]
```

Get the dependency.

#### Returns

VkSubpassDependency const& Vulkan subpass dependency

#### 4.111.2.2 make()

```
static s_ptr lava::subpass_dependency::make (  
    ui32 src_subpass,  
    ui32 dst_subpass,  
    VkDependencyFlags dependency_flags = VK_DEPENDENCY_BY_REGION_BIT) [inline], [static]
```

Make a new subpass dependency.



## Parameters

<i>src_subpass</i>	Source subpass
<i>dst_subpass</i>	Destination subpass
<i>dependency_flags</i>	Dependency flags

## Returns

s\_ptr Shared pointer to subpass dependency

**4.111.2.3 set\_access\_mask()**

```
void lava::subpass_dependency::set_access_mask (
    VkAccessFlags src,
    VkAccessFlags dst) [inline]
```

Set the access mask.

## Parameters

<i>src</i>	Source access flags
<i>dst</i>	Destination access flags

**4.111.2.4 set\_dependency\_flags()**

```
void lava::subpass_dependency::set_dependency_flags (
    VkDependencyFlags flags) [inline]
```

Set the dependency flags.

## Parameters

<i>flags</i>	Dependency flags
--------------	------------------

**4.111.2.5 set\_dst\_access\_mask()**

```
void lava::subpass_dependency::set_dst_access_mask (
    VkAccessFlags mask) [inline]
```

Set the dst access mask.

## Parameters

<i>mask</i>	Access flags
-------------	--------------

**4.111.2.6 set\_dst\_stage\_mask()**

```
void lava::subpass_dependency::set_dst_stage_mask (
    VkPipelineStageFlags mask) [inline]
```

Set the destination stage mask.

**Parameters**

<i>mask</i>	Pipeline stage flags
-------------	----------------------

**4.111.2.7 set\_dst\_subpass()**

```
void lava::subpass_dependency::set_dst_subpass (
    ui32 dst) [inline]
```

Set the dst subpass.

**Parameters**

<i>dst</i>	Destination subpass
------------	---------------------

**4.111.2.8 set\_src\_access\_mask()**

```
void lava::subpass_dependency::set_src_access_mask (
    VkAccessFlags mask) [inline]
```

Set the src access mask.

**Parameters**

<i>mask</i>	Access flags
-------------	--------------

**4.111.2.9 set\_src\_stage\_mask()**

```
void lava::subpass_dependency::set_src_stage_mask (
    VkPipelineStageFlags mask) [inline]
```

Set the source stage mask.

**Parameters**

<i>mask</i>	Pipeline stage flags
-------------	----------------------

**4.111.2.10 set\_src\_subpass()**

```
void lava::subpass_dependency::set_src_subpass (
    ui32 src) [inline]
```

Set the source subpass.

## Parameters

<i>src</i>	Source Subpass
------------	----------------

## 4.111.2.11 set\_stage\_mask()

```
void lava::subpass_dependency::set_stage_mask (
    VkPipelineStageFlags src,
    VkPipelineStageFlags dst) [inline]
```

Set the stage mask.

## Parameters

<i>src</i>	Source pipeline stage flags
<i>dst</i>	Destination pipeline stage flags

## 4.111.2.12 set\_subpass()

```
void lava::subpass_dependency::set_subpass (
    ui32 src,
    ui32 dst) [inline]
```

Set the subpass.

## Parameters

<i>src</i>	Source Subpass
<i>dst</i>	Destination Subpass

The documentation for this struct was generated from the following file:

- [liblava/block/subpass.hpp](#)

## 4.112 lava::surface\_format\_request Struct Reference

Surface format request.

```
#include <format.hpp>
```

## Public Attributes

- [VkFormats formats](#)  
*List of formats in request order.*
- VkColorSpaceKHR **color\_space** = VK\_COLOR\_SPACE\_SRGB\_NONLINEAR\_KHR  
*Color space to request.*

### 4.112.1 Detailed Description

Surface format request.

### 4.112.2 Member Data Documentation

#### 4.112.2.1 formats

`VkFormats` `lava::surface_format_request::formats`

**Initial value:**

```
= {
    VK_FORMAT_B8G8R8A8_UNORM,
    VK_FORMAT_R8G8B8A8_UNORM,
    VK_FORMAT_B8G8R8_UNORM,
    VK_FORMAT_R8G8B8_UNORM,
    VK_FORMAT_B8G8R8A8_SRGB,
    VK_FORMAT_R8G8B8A8_SRGB,
    VK_FORMAT_B8G8R8_SRGB,
    VK_FORMAT_R8G8B8_SRGB,
}
```

List of formats in request order.

The documentation for this struct was generated from the following file:

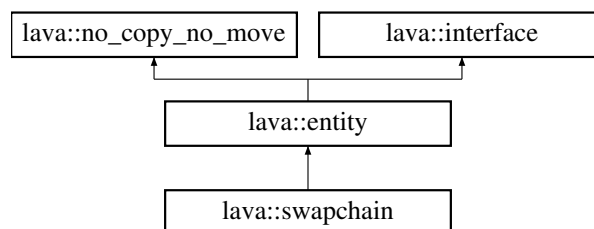
- [liblava/resource/format.hpp](#)

## 4.113 lava::swapchain Struct Reference

Swaphchain.

```
#include <swapchain.hpp>
```

Inheritance diagram for `lava::swapchain`:



### Classes

- struct [callback](#)  
*Swapchain callback.*

**Public Member Functions**

- bool **create** (device::ptr device, VkSurfaceKHR surface, VkSurfaceFormatKHR format, uv2 size, bool v\_sync=false, bool triple\_buffer=true)  
*Create a new swapchain.*
- void **destroy** ()  
*Destroy the swapchain.*
- bool **resize** (uv2 new\_size)  
*Resize the swapchain.*
- void **request\_reload** ()  
*Request a reload of the swapchain.*
- bool **reload\_request** () const  
*Check if reload of the swapchain is requested.*
- device::ptr **get\_device** ()  
*Get the device.*
- uv2 **get\_size** () const  
*Get the size of the swapchain.*
- VkFormat **get\_format** () const  
*Get the format of the swapchain.*
- VkColorSpaceKHR **get\_color\_space** () const  
*Get the color space of the swapchain.*
- VkSwapchainKHR **get** () const  
*Get the swapchain.*
- ui32 **get\_backbuffer\_count** () const  
*Get the backbuffer count.*
- image::s\_list const & **get\_backbuffers** () const  
*Get the backbuffers.*
- void **add\_callback** (callback \*cb)  
*Add callback to swapchain.*
- void **remove\_callback** (callback \*cb)  
*Remove callback from swapchain.*
- bool **v\_sync** () const  
*Check if V-Sync is enabled.*
- bool **triple\_buffer** () const  
*Check if VK\_PRESENT\_MODE\_MAILBOX\_KHR is preferred over VK\_PRESENT\_MODE\_IMMEDIATE\_KHR.*
- bool **surface\_supported** (index queue\_family) const  
*Check if surface is supported by queue family index.*

**Public Member Functions inherited from lava::entity**

- **entity** ()  
*Construct a new entity.*
- id::ref **get\_id** () const  
*Get the id of entity.*

**Public Member Functions inherited from lava::no\_copy\_no\_move**

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** (no\_copy\_no\_move const &)=delete  
*No copy.*
- void **operator=** (no\_copy\_no\_move const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default

*Destroy the interface.*

### 4.113.1 Detailed Description

Swaphchain.

### 4.113.2 Member Function Documentation

#### 4.113.2.1 `add_callback()`

```
void lava::swapchain::add_callback (
    callback * cb)
```

Add callback to swapchain.

##### Parameters

<i>cb</i>	Callback to add
-----------	-----------------

#### 4.113.2.2 `create()`

```
bool lava::swapchain::create (
    device::ptr device,
    VkSurfaceKHR surface,
    VkSurfaceFormatKHR format,
    uv2 size,
    bool v_sync = false,
    bool triple_buffer = true)
```

Create a new swapchain.

##### Parameters

<i>device</i>	Vulkan device
<i>surface</i>	Vulkan surface
<i>format</i>	Surface format
<i>size</i>	Size of swapchain
<i>v_sync</i>	V-Sync enabled
<i>triple_buffer</i>	VK_PRESENT_MODE_MAILBOX_KHR preferred over VK_PRESENT_MODE_IMMEDIATE_KHR

##### Returns

Create was successful or failed

#### 4.113.2.3 get()

```
VkSwapchainKHR lava::swapchain::get () const [inline]
```

Get the swapchain.

##### Returns

VkSwapchainKHR Vulkan swapchain

#### 4.113.2.4 get\_backbuffer\_count()

```
ui32 lava::swapchain::get_backbuffer_count () const [inline]
```

Get the backbuffer count.

##### Returns

ui32 Number of backbuffers

#### 4.113.2.5 get\_backbuffers()

```
image::s_list const & lava::swapchain::get_backbuffers () const [inline]
```

Get the backbuffers.

##### Returns

[image::s\\_list](#) const& List of backbuffer images

#### 4.113.2.6 get\_color\_space()

```
VkColorSpaceKHR lava::swapchain::get_color_space () const [inline]
```

Get the color space of the swapchain.

##### Returns

VkColorSpaceKHR Swapchain color space

#### 4.113.2.7 get\_device()

```
device::ptr lava::swapchain::get_device () [inline]
```

Get the device.

##### Returns

[device::ptr](#) Vulkan device

#### 4.113.2.8 get\_format()

```
VkFormat lava::swapchain::get_format () const [inline]
```

Get the format of the swapchain.

##### Returns

VkFormat Swapchain format

#### 4.113.2.9 get\_size()

```
uv2 lava::swapchain::get_size () const [inline]
```

Get the size of the swapchain.

##### Returns

uv2 Swapchain size

#### 4.113.2.10 reload\_request()

```
bool lava::swapchain::reload_request () const [inline]
```

Check if reload of the swapchain is requested.

##### Returns

Reload is requested or not

#### 4.113.2.11 remove\_callback()

```
void lava::swapchain::remove_callback (  
    callback * cb)
```

Remove callback from swapchain.

##### Parameters

<i>cb</i>	Callback to remove
-----------	--------------------

#### 4.113.2.12 resize()

```
bool lava::swapchain::resize (  
    uv2 new_size)
```

Resize the swapchain.



## Parameters

<i>new_size</i>	New size of swapchain
-----------------	-----------------------

## Returns

Resize was successful or failed

**4.113.2.13 surface\_supported()**

```
bool lava::swapchain::surface_supported (
    index queue_family) const
```

Check if surface is supported by queue family index.

## Parameters

<i>queue_family</i>	Queue family index
---------------------	--------------------

## Returns

Surface is supported by queue family or not

**4.113.2.14 triple\_buffer()**

```
bool lava::swapchain::triple_buffer () const [inline]
```

Check if VK\_PRESENT\_MODE\_MAILBOX\_KHR is preferred over VK\_PRESENT\_MODE\_IMMEDIATE\_KHR.

## Returns

VK\_PRESENT\_MODE\_MAILBOX\_KHR preferred over VK\_PRESENT\_MODE\_IMMEDIATE\_KHR or not

**4.113.2.15 v\_sync()**

```
bool lava::swapchain::v_sync () const [inline]
```

Check if V-Sync is enabled.

## Returns

V-Sync is active or not

The documentation for this struct was generated from the following file:

- [liblava/frame/swapchain.hpp](#)

## 4.114 `lava::target_callback` Struct Reference

Target callback.

```
#include <base.hpp>
```

### Public Types

- using **c\_ptr** = `target_callback` const\*  
*Const pointer to target callback.*
- using **list** = std::vector<`target_callback`\*>  
*List of target callbacks.*
- using **c\_list** = std::vector<`c_ptr`>  
*Const list of target callbacks.*
- using **created\_func** = std::function<bool(`VkAttachmentsRef`, `rect::ref`)>  
*Created function.*
- using **destroyed\_func** = std::function<void()>  
*Destroy function.*

### Public Attributes

- `created_func` **on\_created**  
*Called on target created.*
- `destroyed_func` **on\_destroyed**  
*Called on target destroyed.*

### 4.114.1 Detailed Description

Target callback.

The documentation for this struct was generated from the following file:

- liblava/base/[base.hpp](#)

## 4.115 `lava::telegram` Struct Reference

Telegram.

```
#include <telegram.hpp>
```

### Public Types

- using **ref** = `telegram` const&  
*Reference to telegram.*
- using **set** = std::multiset<`telegram`>  
*Set of telegrams.*

## Public Member Functions

- `telegram` (`id::ref sender`, `id::ref receiver`, `index msg`, `ms dispatch_time={}`, `any info={}`)  
*Construct a new telegram.*
- `bool operator==` (`ref rhs`) `const`  
*Equal operator.*
- `bool operator<` (`ref rhs`) `const`  
*Time order operator.*

## Public Attributes

- `id sender`  
*Sender id.*
- `id receiver`  
*Receiver id.*
- `index msg_id = no_index`  
*Message id.*
- `ms dispatch_time`  
*Dispatch time.*
- `any info`  
*Telegram information.*

## 4.115.1 Detailed Description

Telegram.

## 4.115.2 Constructor & Destructor Documentation

### 4.115.2.1 telegram()

```

lava::telegram::telegram (
    id::ref sender,
    id::ref receiver,
    index msg,
    ms dispatch_time = {},
    any info = {}) [inline], [explicit]

```

Construct a new telegram.

#### Parameters

<i>sender</i>	Sender id
<i>receiver</i>	Receiver id
<i>msg</i>	Message id
<i>dispatch_time</i>	Dispatch time
<i>info</i>	Telegram information

## 4.115.3 Member Function Documentation

### 4.115.3.1 operator<()

```

bool lava::telegram::operator< (
    ref rhs) const [inline]

```

Time order operator.

## Parameters

<i>rhs</i>	Another telegram
------------	------------------

## Returns

Telegram is earlier or later

## 4.115.3.2 operator==()

```
bool lava::telegram::operator== (
    ref rhs) const [inline]
```

Equal operator.

## Parameters

<i>rhs</i>	Another telegram
------------	------------------

## Returns

Telegram is equal or not

The documentation for this struct was generated from the following file:

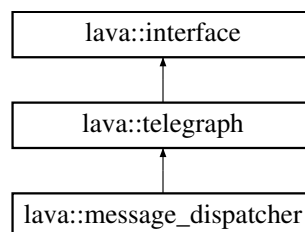
- [liblava/util/telegram.hpp](#)

## 4.116 lava::telegraph Struct Reference

Telegraph station.

```
#include <telegram.hpp>
```

Inheritance diagram for lava::telegraph:



## Public Member Functions

- virtual void [send\\_message](#) (id::ref receiver, id::ref sender, index message, ms delay={}, any const &info={})=0  
*Send message to dispatcher.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default  
*Destroy the interface.*

### 4.116.1 Detailed Description

Telegraph station.

### 4.116.2 Member Function Documentation

#### 4.116.2.1 `send_message()`

```
virtual void lava::telegraph::send_message (
    id::ref receiver,
    id::ref sender,
    index message,
    ms delay = {},
    any const & info = {}) [pure virtual]
```

Send message to dispatcher.

#### Parameters

<i>receiver</i>	Receiver id
<i>sender</i>	Sender id
<i>message</i>	Message id
<i>delay</i>	Delay time
<i>info</i>	Telegram information

Implemented in [lava::message\\_dispatcher](#).

The documentation for this struct was generated from the following file:

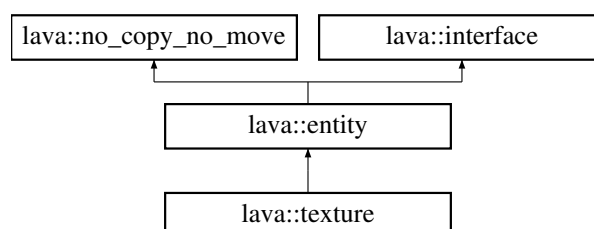
- [liblava/util/telegram.hpp](#)

## 4.117 lava::texture Struct Reference

Texture.

```
#include <texture.hpp>
```

Inheritance diagram for `lava::texture`:



## Classes

- struct [layer](#)  
*Texture layer.*
- struct [mip\\_level](#)  
*Texture mip level.*

## Public Types

- using [s\\_ptr](#) = std::shared\_ptr<[texture](#)>  
*Shared pointer to texture.*
- using [s\\_map](#) = std::map<[id](#), [s\\_ptr](#)>  
*Map of textures.*
- using [s\\_list](#) = std::vector<[s\\_ptr](#)>  
*List of textures.*

## Public Member Functions

- [~texture](#) ()  
*Destroy the texture.*
- bool [create](#) ([device::ptr](#) device, [uv2](#) size, VkFormat format, [layer::list](#) const &layers={}, [texture\\_type](#) type=[texture\\_type::tex\\_2d](#))  
*Create a new texture.*
- void [destroy](#) ()  
*Destroy the texture.*
- bool [upload](#) (void const \*[data](#), [size\\_t](#) data\_size)  
*Upload data to texture.*
- bool [stage](#) (VkCommandBuffer cmd\_buffer)  
*Stage the texture.*
- void [destroy\\_upload\\_buffer](#) ()  
*Destroy the upload buffer.*
- VkDescriptorImageInfo const \* [get\\_descriptor\\_info](#) () const  
*Get the descriptor information.*
- [image::s\\_ptr](#) [get\\_image](#) ()  
*Get the image of the texture.*
- [uv2](#) [get\\_size](#) () const  
*Get the size of the texture.*
- [texture\\_type](#) [get\\_type](#) () const  
*Get the type of the texture.*
- VkFormat [get\\_format](#) () const  
*Get the format of the texture.*

## Public Member Functions inherited from [lava::entity](#)

- [entity](#) ()  
*Construct a new entity.*
- [id::ref](#) [get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- **no\_copy\_no\_move** ()=default  
*Construct a new object.*
- **no\_copy\_no\_move** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void **operator=** ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual **~interface** ()=default  
*Destroy the interface.*

## Static Public Member Functions

- static [s\\_ptr](#) **make** ()  
*Make a new texture.*

### 4.117.1 Detailed Description

Texture.

### 4.117.2 Member Function Documentation

#### 4.117.2.1 create()

```
bool lava::texture::create (
    device::ptr device,
    uv2 size,
    VkFormat format,
    layer::list const & layers = {},
    texture\_type type = texture_type::tex_2d)
```

Create a new texture.

#### Parameters

<i>device</i>	Vulkan device
<i>size</i>	Texture size
<i>format</i>	Texture format
<i>layers</i>	List of layers
<i>type</i>	Texture type

#### Returns

Create was successful or failed

#### 4.117.2.2 `get_descriptor_info()`

```
VkDescriptorImageInfo const * lava::texture::get_descriptor_info () const [inline]
```

Get the descriptor information.

##### Returns

VkDescriptorImageInfo const\* Descriptor image information

#### 4.117.2.3 `get_format()`

```
VkFormat lava::texture::get_format () const [inline]
```

Get the format of the texture.

##### Returns

VkFormat Texture format

#### 4.117.2.4 `get_image()`

```
image::s_ptr lava::texture::get_image () [inline]
```

Get the image of the texture.

##### Returns

image::s\_ptr Shared pointer to image

#### 4.117.2.5 `get_size()`

```
uv2 lava::texture::get_size () const [inline]
```

Get the size of the texture.

##### Returns

uv2 Texture size

#### 4.117.2.6 `get_type()`

```
texture_type lava::texture::get_type () const [inline]
```

Get the type of the texture.

##### Returns

texture\_type Texture type



#### 4.117.2.7 make()

```
static s_ptr lava::texture::make () [inline], [static]
```

Make a new texture.

##### Returns

s\_ptr Shared pointer to texture

#### 4.117.2.8 stage()

```
bool lava::texture::stage (  
    VkCommandBuffer cmd_buffer)
```

Stage the texture.

##### Parameters

<i>cmd_buffer</i>	Command buffer
-------------------	----------------

##### Returns

Stage was successful or failed

#### 4.117.2.9 upload()

```
bool lava::texture::upload (  
    void const * data,  
    size_t data_size)
```

Upload data to texture.

##### Parameters

<i>data</i>	Data to upload
<i>data_size</i>	Size of data

##### Returns

Upload was successful or failed

The documentation for this struct was generated from the following file:

- [liblava/resource/texture.hpp](#)

## 4.118 `lava::texture_file` Struct Reference

Texture file path with format.

```
#include <texture.hpp>
```

### Public Types

- using **list** = `std::vector<texture\_file>`  
*List of texture files.*

### Public Attributes

- [string](#) **path**  
*File path.*
- `VkFormat` **format** = `VK_FORMAT_UNDEFINED`  
*File format.*

### 4.118.1 Detailed Description

Texture file path with format.

The documentation for this struct was generated from the following file:

- `liblava/resource/texture.hpp`

## 4.119 `lava::thread_pool` Struct Reference

Thread pool.

```
#include <thread.hpp>
```

### Public Types

- using **task** = `std::function<void(id::ref)>`  
*Task function (with thread id)*

### Public Member Functions

- void [setup](#) ([ui32](#) count=2)  
*Set up the thread pool.*
- void **teardown** ()  
*Tear down the thread pool.*
- void [enqueue](#) (auto f)  
*Enqueue a task.*

### 4.119.1 Detailed Description

Thread pool.

### 4.119.2 Member Function Documentation

#### 4.119.2.1 enqueue()

```
void lava::thread_pool::enqueue (
    auto f) [inline]
```

Enqueue a task.

##### Parameters

<i>f</i>	Task function
----------	---------------

#### 4.119.2.2 setup()

```
void lava::thread_pool::setup (
    ui32 count = 2) [inline]
```

Set up the thread pool.

##### Parameters

<i>count</i>	Number of threads
--------------	-------------------

The documentation for this struct was generated from the following file:

- [liblava/util/thread.hpp](#)

## 4.120 lava::timer Struct Reference

Timer.

```
#include <time.hpp>
```

### Public Member Functions

- **timer** ()  
*Construct a new timer.*
- void **reset** ()  
*Reset the timer.*
- **ms elapsed** () const  
*Get the elapsed time.*

### 4.120.1 Detailed Description

Timer.

### 4.120.2 Member Function Documentation

#### 4.120.2.1 elapsed()

```
ms lava::timer::elapsed () const [inline]
```

Get the elapsed time.

Returns

ms Elapsed milliseconds

The documentation for this struct was generated from the following file:

- [liblava/core/time.hpp](#)

## 4.121 lava::tooltip Struct Reference

Tooltip.

```
#include <input.hpp>
```

### Public Types

- using **list** = std::vector<[tooltip](#)>  
*List of tooltips.*

### Public Member Functions

- [tooltip](#) (string\_ref name, key key, mod mod)  
*Construct a new tooltip.*

### Public Attributes

- [string](#) name  
*Name of tooltip.*
- [lava::key](#) key  
*Input key.*
- [lava::mod](#) mod  
*Input mod.*

### 4.121.1 Detailed Description

Tooltip.

### 4.121.2 Constructor & Destructor Documentation

#### 4.121.2.1 `tooltip()`

```
lava::tooltip::tooltip (  
    string\_ref name,  
    key key,  
    mod mod) [inline]
```

Construct a new tooltip.

#### Parameters

<i>name</i>	Name of tooltip
<i>key</i>	Input key
<i>mod</i>	Input mod

The documentation for this struct was generated from the following file:

- [liblava/frame/input.hpp](#)

## 4.122 `lava::tooltip_list` Struct Reference

Tooltip list.

```
#include <input.hpp>
```

#### Public Member Functions

- void [add](#) ([string\\_ref](#) name, [key](#) key, [mod](#) mod=`mod::none`)  
*Add a tooltip.*
- void [clear](#) ()  
*Clear tooltips.*
- [tooltip::list](#) const & [get\\_list](#) () const  
*Get tooltips.*
- void [set](#) ([tooltip::list](#) const &list)  
*Set a new tooltip list.*
- [string](#) [format\\_string](#) () const  
*Convert tooltips to string.*

### 4.122.1 Detailed Description

Tooltip list.

## 4.122.2 Member Function Documentation

### 4.122.2.1 add()

```
void lava::tooltip_list::add (  
    string_ref name,  
    key key,  
    mod mod = mod::none) [inline]
```

Add a tooltip.

#### Parameters

<i>name</i>	Name of tooltip
<i>key</i>	Input key
<i>mod</i>	Input mod (default: none)

### 4.122.2.2 format\_string()

```
string lava::tooltip_list::format_string () const
```

Convert tooltips to string.

#### Returns

string String representation

### 4.122.2.3 get\_list()

```
tooltip::list const & lava::tooltip_list::get_list () const [inline]
```

Get tooltips.

#### Returns

[tooltip::list](#) List of tooltips

### 4.122.2.4 set()

```
void lava::tooltip_list::set (  
    tooltip::list const & list) [inline]
```

Set a new tooltip list.

#### Parameters

<i>list</i>	List of tooltips
-------------	------------------

The documentation for this struct was generated from the following file:

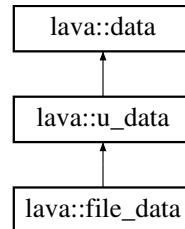
- [liblava/frame/input.hpp](#)

## 4.123 `lava::u_data` Struct Reference

Unique data wrapper.

```
#include <data.hpp>
```

Inheritance diagram for `lava::u_data`:



### Public Types

- using **ref** = `u_data` const&  
*Reference to unique data wrapper.*

### Public Types inherited from `lava::data`

- enum class **mode** : index { **alloc** = 0 , **no\_alloc** }  
*Data modes.*
- using **ref** = `data` const&  
*Reference to data wrapper.*
- using **ptr** = char\*  
*Data pointer.*
- using **c\_ptr** = char const\*  
*Const data pointer.*

### Public Member Functions

- `u_data` (`size_t` length=0, `data::mode` mode=`data::mode::alloc`)  
*Construct a new unique data.*
- `u_data` (`data::ref` data)  
*Construct a new unique data from another data.*
- `~u_data` ()  
*Destroy the unique data.*

### Public Member Functions inherited from `lava::data`

- `data` ()=default  
*Construct a new data.*
- `data` (auto \*addr, `size_t` size)  
*Construct a new data.*
- bool `set` (`size_t` length, `mode` mode=`mode::alloc`)  
*Set and allocate data by length.*
- bool `allocate` ()  
*Allocate data.*
- void `deallocate` ()  
*Deallocate data.*
- `ptr` `end` () const  
*Pointer to end of data.*

## Additional Inherited Members

### Static Public Member Functions inherited from [lava::data](#)

- static [ptr](#) [as\\_ptr](#) (auto \*value)  
*Cast to data pointer.*
- static [c\\_ptr](#) [as\\_c\\_ptr](#) (auto \*value)  
*Cast to const data pointer.*

### Public Attributes inherited from [lava::data](#)

- [ptr](#) [addr](#) = nullptr  
*Pointer address.*
- [size\\_t](#) [size](#) = 0  
*Size of data.*
- [size\\_t](#) [alignment](#) = 0  
*Data alignment.*

## 4.123.1 Detailed Description

Unique data wrapper.

## 4.123.2 Constructor & Destructor Documentation

### 4.123.2.1 [u\\_data\(\)](#) [1/2]

```
lava::u_data::u_data (
    size\_t length = 0,
    data::mode mode = data::mode::alloc) [inline]
```

Construct a new unique data.

#### Parameters

<i>length</i>	Length of data
<i>mode</i>	Data mode

### 4.123.2.2 [u\\_data\(\)](#) [2/2]

```
lava::u_data::u_data (
    data::ref data) [inline], [explicit]
```

Construct a new unique data from another data.

#### Parameters

<i>data</i>	Source data
-------------	-------------

The documentation for this struct was generated from the following file:

- [liblava/core/data.hpp](#)



## 4.124 lava::version Struct Reference

Version.

```
#include <version.hpp>
```

### Public Attributes

- **ui32** **year** = 2024  
*Version year.*
- **ui32** **release** = 0  
*Version release.*
- **version\_stage** **stage** = version\_stage::rolling  
*Version stage.*
- **ui32** **rev** = 0  
*Version revision.*

### 4.124.1 Detailed Description

Version.

The documentation for this struct was generated from the following file:

- liblava/core/[version.hpp](#)

## 4.125 lava::vertex Struct Reference

Vertex.

```
#include <primitive.hpp>
```

### Public Types

- using **list** = std::vector<[vertex](#)>  
*List of vertices.*

### Public Member Functions

- bool [operator==](#) ([vertex](#) const &other) const  
*Equal compare operator.*

## Public Attributes

- **v3 position**  
*Vertex position.*
- **v4 color**  
*Vertex color.*
- **v2 uv**  
*Vertex uv.*
- **v3 normal**  
*Vertex normal.*

### 4.125.1 Detailed Description

Vertex.

### 4.125.2 Member Function Documentation

#### 4.125.2.1 operator==( )

```
bool lava::vertex::operator==(
    vertex const & other) const [inline]
```

Equal compare operator.

#### Parameters

<i>other</i>	Another vertex
--------------	----------------

#### Returns

Another vertex is equal or not

The documentation for this struct was generated from the following file:

- [liblava/resource/primitive.hpp](#)

## 4.126 lava::vk\_result Struct Reference

Vulkan result.

```
#include <base.hpp>
```

## Public Member Functions

- **operator bool ( )**  
*Check result.*

## Public Attributes

- bool **state** = false  
*State of result.*
- VkResult **value** = VK\_NOT\_READY  
*Value of result.*

### 4.126.1 Detailed Description

Vulkan result.

### 4.126.2 Member Function Documentation

#### 4.126.2.1 operator bool()

```
lava::vk_result::operator bool () [inline]
```

Check result.

#### Returns

Okay or error

The documentation for this struct was generated from the following file:

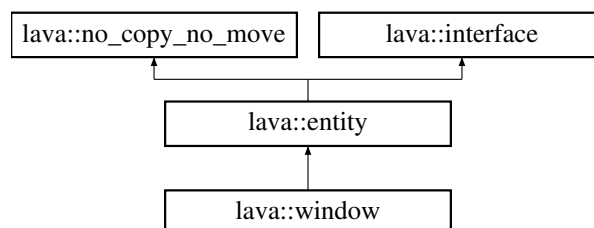
- liblava/base/[base.hpp](#)

## 4.127 lava::window Struct Reference

Window.

```
#include <window.hpp>
```

Inheritance diagram for lava::window:



## Classes

- struct [state](#)  
*Window state.*

## Public Types

- using **ptr** = [window](#)\*  
*Pointer to window.*
- using **s\_ptr** = std::shared\_ptr<[window](#)>  
*Shared pointer to window.*
- using **event** = std::function<void([s\\_ptr](#))>  
*Window event function.*
- using **s\_map** = std::map<[id](#), [s\\_ptr](#)>  
*Map of windows.*
- using **ref** = [window](#) const&  
*Reference to window.*
- using **resize\_func** = std::function<bool([ui32](#), [ui32](#))>  
*Resize window function.*

## Public Member Functions

- **window** ()=default  
*Construct a new window.*
- [window](#) ([name](#) title)  
*Construct a new window.*
- bool [create](#) ([state::optional](#) state={})  
*Create a new window with optional state.*
- void **destroy** ()  
*Destroy the window.*
- [state](#) [get\\_state](#) () const  
*Get the window state.*
- void [set\\_state](#) ([state](#) &s)  
*Set the window state.*
- void [set\\_title](#) ([string\\_ref](#) text)  
*Set the window title.*
- [string\\_ref](#) [get\\_title](#) () const  
*Get the window title.*
- void [set\\_save\\_name](#) ([string\\_ref](#) save)  
*Set the save name.*
- [string\\_ref](#) [get\\_save\\_name](#) () const  
*Get the save name.*
- void [set\\_position](#) ([i32](#) x, [i32](#) y)  
*Set the position of window.*
- void [get\\_position](#) ([i32](#) &x, [i32](#) &y) const  
*Get the position of window.*
- void [set\\_size](#) ([ui32](#) width, [ui32](#) height)  
*Set the size of window.*
- void [get\\_size](#) ([ui32](#) &width, [ui32](#) &height) const  
*Get the size of window.*
- void [get\\_framebuffer\\_size](#) ([ui32](#) &width, [ui32](#) &height) const  
*Get the framebuffer size.*
- [uv2](#) [get\\_size](#) () const  
*Get the size.*
- [uv2](#) [get\\_framebuffer\\_size](#) () const

- Get the framebuffer size.*

  - void `set_mouse_position` (r64 x, r64 y)
- Set the mouse position.*

  - void `get_mouse_position` (r64 &x, r64 &y) const
- Get the mouse position.*

  - `v2 get_content_scale` () const
- Get the content scale.*

  - `mouse_position get_mouse_position` () const
- Get the mouse position in window.*

  - void `hide_mouse_cursor` ()
- Hide mouse cursor.*

  - void `show_mouse_cursor` ()
- Show mouse cursor.*

  - `r32 get_aspect_ratio` () const
- Get the aspect ratio of window.*

  - void `show` ()
- Show the window.*

  - void `hide` ()
- Hide the window.*

  - bool `visible` () const
- Check if window is visible.*

  - void `iconify` ()
- Iconify the window.*

  - bool `iconified` () const
- Check if the window is iconified.*

  - void `restore` ()
- Restore the window.*

  - void `maximize` ()
- Maximize the window.*

  - bool `maximized` () const
- Check if the window is maximized.*

  - void `focus` ()
- Focus the window.*

  - bool `focused` () const
- Check if the window is focused.*

  - void `set_fullscreen` (bool active)
- Set the window to fullscreen.*

  - bool `fullscreen` () const
- Check if the window is fullscreen.*

  - bool `hovered` () const
- Check if mouse hovered over the window.*

  - bool `resizable` () const
- Check if the window is resizable.*

  - void `set_resizable` (bool value)
- Set the window resizable.*

  - bool `decorated` () const
- Check if the window is decorated.*

  - void `set_decorated` (bool value)
- Set the window decorated.*

  - bool `floating` () const
- Check if the window is floating.*

- void [set\\_floating](#) (bool value)  
*Set the window floating.*
- bool [close\\_request](#) () const  
*Check if the window request to close.*
- bool [switch\\_mode\\_request](#) () const  
*Check if the window request to switch mode.*
- bool [switch\\_mode](#) (state::optional state={})  
*Switch mode of the window.*
- GLFWwindow \* [get](#) () const  
*Get GLFW handle.*
- bool [resize\\_request](#) () const  
*Check if the window request to resize.*
- bool [handle\\_resize](#) ()  
*Handle window resize.*
- void [update\\_state](#) ()  
*Update window state.*
- void [assign](#) (input::ptr callback)  
*Assign input callback.*
- void [show\\_save\\_title](#) (bool value=true)  
*Show the save title in the window.*
- bool [save\\_title](#) () const  
*Check the show save title state.*
- void [update\\_title](#) ()  
*Update the window title.*
- VkSurfaceKHR [create\\_surface](#) ()  
*Create a surface.*
- void [set\\_icon](#) (data::c\_ptr data, uv2 size)  
*Set the window icon.*
- [index\\_detect\\_monitor](#) () const  
*Detect the monitor index of the window.*
- void [center](#) ()  
*Center the window on the monitor.*

## Public Member Functions inherited from [lava::entity](#)

- [entity](#) ()  
*Construct a new entity.*
- [id::ref get\\_id](#) () const  
*Get the id of entity.*

## Public Member Functions inherited from [lava::no\\_copy\\_no\\_move](#)

- [no\\_copy\\_no\\_move](#) ()=default  
*Construct a new object.*
- [no\\_copy\\_no\\_move](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No copy.*
- void [operator=](#) ([no\\_copy\\_no\\_move](#) const &)=delete  
*No move.*

## Public Member Functions inherited from [lava::interface](#)

- virtual `~interface()`=default  
*Destroy the interface.*

## Public Attributes

- [resize\\_func](#) `on_resize`  
*Called on window resize.*

### 4.127.1 Detailed Description

Window.

### 4.127.2 Constructor & Destructor Documentation

#### 4.127.2.1 `window()`

```
lava::window::window (
    name title) [inline], [explicit]
```

Construct a new window.

#### Parameters

<i>title</i>	Title of window
--------------	-----------------

### 4.127.3 Member Function Documentation

#### 4.127.3.1 `assign()`

```
void lava::window::assign (
    input::ptr callback) [inline]
```

Assign input callback.

#### Parameters

<i>callback</i>	Input callbacl
-----------------	----------------

#### 4.127.3.2 `close_request()`

```
bool lava::window::close_request () const
```

Check if the window request to close.

#### Returns

Window has close request or not

#### 4.127.3.3 `create()`

```
bool lava::window::create (
    state::optional state = {})
```

Create a new window with optional state.

**Parameters**

<i>state</i>	Window state
--------------	--------------

**Returns**

Create was successful or failed

**4.127.3.4 create\_surface()**

```
VkSurfaceKHR lava::window::create_surface ()
```

Create a surface.

**Returns**

VkSurfaceKHR Vulkan surface

**4.127.3.5 decorated()**

```
bool lava::window::decorated () const
```

Check if the window is decorated.

**Returns**

Window is decorated or not

**4.127.3.6 detect\_monitor()**

```
index lava::window::detect_monitor () const
```

Detect the monitor index of the window.

**Returns**

index Monitor index

**4.127.3.7 floating()**

```
bool lava::window::floating () const
```

Check if the window is floating.

**Returns**

Window is floating or not



#### 4.127.3.8 focused()

```
bool lava::window::focused () const
```

Check if the window is focused.

##### Returns

Window is focused or not

#### 4.127.3.9 fullscreen()

```
bool lava::window::fullscreen () const [inline]
```

Check if the window is fullscreen.

##### Returns

Window is fullscreen or not

#### 4.127.3.10 get()

```
GLFWwindow * lava::window::get () const [inline]
```

Get GLFW handle.

##### Returns

GLFWwindow\* GLFW window handle

#### 4.127.3.11 get\_aspect\_ratio()

```
r32 lava::window::get_aspect_ratio () const
```

Get the aspect ratio of window.

##### Returns

r32 Aspect ratio

#### 4.127.3.12 get\_content\_scale()

```
v2 lava::window::get_content_scale () const
```

Get the content scale.

##### Returns

v2 Window content scale

**4.127.3.13 get\_framebuffer\_size()** [1/2]

```
uv2 lava::window::get_framebuffer_size () const
```

Get the framebuffer size.

**Returns**

uv2 Size of framebuffer

**4.127.3.14 get\_framebuffer\_size()** [2/2]

```
void lava::window::get_framebuffer_size (
    ui32 & width,
    ui32 & height) const
```

Get the framebuffer size.

**Parameters**

<i>width</i>	Framebuffer width
<i>height</i>	Framebuffer height

**4.127.3.15 get\_mouse\_position()** [1/2]

```
mouse_position lava::window::get_mouse_position () const
```

Get the mouse position in window.

**Returns**

[mouse\\_position](#) Position of mouse

**4.127.3.16 get\_mouse\_position()** [2/2]

```
void lava::window::get_mouse_position (
    r64 & x,
    r64 & y) const
```

Get the mouse position.

**Parameters**

<i>x</i>	Mouse X position
<i>y</i>	Mouse Y position

**4.127.3.17 get\_position()**

```
void lava::window::get_position (
    i32 & x,
    i32 & y) const
```

Get the position of window.

## Parameters

<i>x</i>	X position
<i>y</i>	Y position

**4.127.3.18 get\_save\_name()**

```
string_ref lava::window::get_save_name () const [inline]
```

Get the save name.

## Returns

name Save name of window

**4.127.3.19 get\_size() [1/2]**

```
uv2 lava::window::get_size () const
```

Get the size.

## Returns

uv2 Size of window

**4.127.3.20 get\_size() [2/2]**

```
void lava::window::get_size (
    ui32 & width,
    ui32 & height) const
```

Get the size of window.

## Parameters

<i>width</i>	Window width
<i>height</i>	Window height

**4.127.3.21 get\_state()**

```
state lava::window::get_state () const
```

Get the window state.

## Returns

state Window state

#### 4.127.3.22 `get_title()`

```
string_ref lava::window::get_title () const [inline]
```

Get the window title.

##### Returns

name Title of window

#### 4.127.3.23 `handle_resize()`

```
bool lava::window::handle_resize () [inline]
```

Handle window resize.

##### Returns

Resize was successful or failed

#### 4.127.3.24 `hovered()`

```
bool lava::window::hovered () const
```

Check if mouse hovered over the window.

##### Returns

Mouse hovered or not

#### 4.127.3.25 `iconified()`

```
bool lava::window::iconified () const
```

Check if the window is iconified.

##### Returns

Window is iconified or not

#### 4.127.3.26 `maximized()`

```
bool lava::window::maximized () const
```

Check if the window is maximized.

##### Returns

Window is maximized or not

#### 4.127.3.27 resizable()

```
bool lava::window::resizable () const
```

Check if the window is resizable.

##### Returns

Window is resizable or not

#### 4.127.3.28 resize\_request()

```
bool lava::window::resize_request () const [inline]
```

Check if the window request to resize.

##### Returns

Window has resize request or not

#### 4.127.3.29 save\_title()

```
bool lava::window::save_title () const [inline]
```

Check the show save title state.

##### Returns

Save title is active or not

#### 4.127.3.30 set\_decorated()

```
void lava::window::set_decorated (  
    bool value)
```

Set the window decorated.

##### Parameters

<i>value</i>	Decorated state
--------------	-----------------

#### 4.127.3.31 set\_floating()

```
void lava::window::set_floating (  
    bool value)
```

Set the window floating.

## Parameters

<i>value</i>	Floating state
--------------	----------------

**4.127.3.32 set\_fullscreen()**

```
void lava::window::set_fullscreen (  
    bool active) [inline]
```

Set the window to fullscreen.

## Parameters

<i>active</i>	Fullscreen or windowed mode
---------------	-----------------------------

**4.127.3.33 set\_icon()**

```
void lava::window::set_icon (  
    data::c_ptr data,  
    uv2 size)
```

Set the window icon.

## Parameters

<i>data</i>	Image data
<i>size</i>	Image size

**4.127.3.34 set\_mouse\_position()**

```
void lava::window::set_mouse_position (  
    r64 x,  
    r64 y)
```

Set the mouse position.

## Parameters

<i>x</i>	Mouse X position
<i>y</i>	Mouse Y position

**4.127.3.35 set\_position()**

```
void lava::window::set_position (  
    i32 x,  
    i32 y)
```

Set the position of window.

## Parameters

<i>x</i>	X positoin
<i>y</i>	Y position

**4.127.3.36 set\_resizable()**

```
void lava::window::set_resizable (  
    bool value)
```

Set the window resizable.

## Parameters

<i>value</i>	Resizable state
--------------	-----------------

**4.127.3.37 set\_save\_name()**

```
void lava::window::set_save_name (  
    string_ref save) [inline]
```

Set the save name.

## Parameters

<i>save</i>	Save name of window
-------------	---------------------

**4.127.3.38 set\_size()**

```
void lava::window::set_size (  
    ui32 width,  
    ui32 height)
```

Set the size of window.

## Parameters

<i>width</i>	Window width
<i>height</i>	Window height

**4.127.3.39 set\_state()**

```
void lava::window::set_state (  
    state & s)
```

Set the window state.

## Parameters

<i>s</i>	Window state
----------	--------------

**4.127.3.40 set\_title()**

```
void lava::window::set_title (  
    string_ref text)
```

Set the window title.

## Parameters

<i>text</i>	Title of window
-------------	-----------------

**4.127.3.41 show\_save\_title()**

```
void lava::window::show_save_title (  
    bool value = true) [inline]
```

Show the save title in the window.

## Parameters

<i>value</i>	Save title state
--------------	------------------

**4.127.3.42 switch\_mode()**

```
bool lava::window::switch_mode (  
    state::optional state = {})
```

Switch mode of the window.

## Parameters

<i>state</i>	Target window state
--------------	---------------------

## Returns

Switch was successful or failed

**4.127.3.43 switch\_mode\_request()**

```
bool lava::window::switch_mode_request () const [inline]
```

Check if the window request to switch mode.

## Returns

Window has switch mode request or not



#### 4.127.3.44 visible()

```
bool lava::window::visible () const
```

Check if window is visible.

##### Returns

Window is visible or not

The documentation for this struct was generated from the following file:

- liblava/frame/[window.hpp](#)



## Chapter 5

# File Documentation

### 5.1 liblava/app.hpp File Reference

App module.

```
#include "liblava/app/app.hpp"
#include "liblava/app/benchmark.hpp"
#include "liblava/app/camera.hpp"
#include "liblava/app/config.hpp"
#include "liblava/app/def.hpp"
#include "liblava/app/forward_shading.hpp"
#include "liblava/app/imgui.hpp"
```

#### 5.1.1 Detailed Description

App module.

##### Author

Lava Block OÜ and contributors

##### Copyright

Copyright (c) 2018-present, MIT License

### 5.2 app.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/app/app.hpp"
00011 #include "liblava/app/benchmark.hpp"
00012 #include "liblava/app/camera.hpp"
00013 #include "liblava/app/config.hpp"
00014 #include "liblava/app/def.hpp"
00015 #include "liblava/app/forward_shading.hpp"
00016 #include "liblava/app/imgui.hpp"
```

## 5.3 liblava/app/app.hpp File Reference

Application with basic functionality.

```
#include "liblava/app/benchmark.hpp"
#include "liblava/app/camera.hpp"
#include "liblava/app/config.hpp"
#include "liblava/app/forward_shading.hpp"
#include "liblava/block.hpp"
#include "liblava/frame.hpp"
```

### Classes

- struct [lava::app](#)  
*Application with basic functionality.*
- struct [lava::app::about\\_info\\_setting](#)

### 5.3.1 Detailed Description

Application with basic functionality.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.4 app.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/app/benchmark.hpp"
00011 #include "liblava/app/camera.hpp"
00012 #include "liblava/app/config.hpp"
00013 #include "liblava/app/forward_shading.hpp"
00014 #include "liblava/block.hpp"
00015 #include "liblava/frame.hpp"
00016
00017 namespace lava {
00018
00022 struct app : frame {
00027     explicit app(frame_env::ref env);
00028
00034     explicit app(name name, argh::parser cmd_line = {});
00035
00040     virtual bool setup();
00041
00048     bool headless = false;
00049
00051     lava::window window;
00052
00054     lava::input input;
00055
00057     lava::imgui imgui;
00058
```

```

00060     ImGui::config ImGui_config;
00061
00063     tooltip_list tooltips;
00064
00066     lava::device::ptr device = nullptr;
00067
00069     lava::camera camera;
00070
00072     gamepad pad;
00073
00075     lava::staging staging;
00076
00078     lava::block block;
00079
00081     lava::renderer renderer;
00082
00084     forward_shading shading;
00085
00087     render_target::s_ptr target;
00088
00090     file_system fs;
00091
00093     VkPipelineCache pipeline_cache = nullptr;
00094
00096     using update_func = std::function<bool(delta)>;
00097
00099     update_func on_update;
00100
00102     using create_func = std::function<bool()>;
00103
00105     create_func on_create;
00106
00108     using destroy_func = std::function<void()>;
00109
00111     destroy_func on_destroy;
00112
00117     bool v_sync() const {
00118         return config.v_sync;
00119     }
00120
00125     bool triple_buffer() const {
00126         return config.triple_buffer;
00127     }
00128
00133     ui32 fps_cap() const {
00134         return config.fps_cap;
00135     }
00136
00141     ui32 get_frame_counter() const {
00142         return m_frame_counter;
00143     }
00144
00149     string get_fps_info() const;
00150
00154     struct about_info_setting {
00156         bool draw_separator = true;
00157
00159         bool draw_fps = true;
00160
00162         bool draw_spacing = true;
00163
00168         static about_info_setting all() {
00169             return {};
00170         }
00171     };
00172
00177     void draw_about(about_info_setting setting = about_info_setting::all()) const;
00178
00180     app_config config;
00181
00183     json_file config_file;
00184
00186     using process_func = std::function<void(VkCommandBuffer, index)>;
00187
00189     process_func on_process;
00190
00195     id::ref block_cmd() const {
00196         return m_block_command;
00197     }
00198
00200     using setup_func = std::function<bool()>;
00201
00203     setup_func on_setup;
00204
00209     string screenshot();
00210
00215     void switch_config(string_ref config_name);

```

```

00216
00217 private:
00221     void mount_resource();
00222
00227     bool setup_file_system();
00228
00233     bool setup_window();
00234
00239     bool setup_device();
00240
00245     bool setup_render();
00246
00250     void setup_run();
00251
00255     void parse_cmd_line();
00256
00262     bool load_config(string_ref config_name);
00263
00267     void handle_input();
00268
00272     void handle_keys();
00273
00277     void handle_window();
00278
00282     void update();
00283
00287     void render();
00288
00293     bool create_imgui();
00294
00298     void destroy_imgui();
00299
00304     bool create_target();
00305
00309     void destroy_target();
00310
00315     bool create_block();
00316
00321     bool create_pipeline_cache();
00322
00326     void destroy_pipeline_cache();
00327
00329     texture::s_ptr m_imgui_fonts;
00330
00332     bool m_toggle_v_sync = false;
00333
00335     ui32 m_frame_counter = 0;
00336
00338     us m_last_render_time{0};
00339
00341     json_file::callback m_config_callback;
00342
00344     id m_block_command;
00345
00347     benchmark_data m_frames;
00348 };
00349
00350 } // namespace lava

```

## 5.5 liblava/app/benchmark.hpp File Reference

Benchmark.

```

#include "liblava/app/def.hpp"
#include "liblava/frame/frame.hpp"

```

### Classes

- struct [lava::benchmark\\_data](#)  
*Benchmark data.*

## Functions

- bool `lava::parse_benchmark` (`cmd_line cmd_line`, `benchmark_data &data`)  
*Parse command line arguments and set benchmark data.*
- void `lava::benchmark` (`frame &app`, `benchmark_data &data`)  
*Start a benchmark run.*
- bool `lava::write_frames_json` (`benchmark_data &data`)  
*Write frames to json file.*

### 5.5.1 Detailed Description

Benchmark.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.5.2 Function Documentation

#### 5.5.2.1 `benchmark()`

```
void lava::benchmark (  
    frame & app,  
    benchmark_data & data)
```

Start a benchmark run.

##### Parameters

<code>app</code>	App to benchmark
<code>data</code>	Benchmark data setting

#### 5.5.2.2 `parse_benchmark()`

```
bool lava::parse_benchmark (  
    cmd_line cmd_line,  
    benchmark_data & data)
```

Parse command line arguments and set benchmark data.

##### Parameters

<code>cmd_line</code>	Command line arguments
<code>data</code>	Benchmark data

##### Returns

Benchmark data is parsed or not ready

### 5.5.2.3 write\_frames\_json()

```
bool lava::write_frames_json (
    benchmark_data & data)
```

Write frames to json file.



## Parameters

<i>data</i>	Benchmark data setting
-------------	------------------------

## Returns

Write was successful or failed

## 5.6 benchmark.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/app/def.hpp"
00011 #include "liblava/frame/frame.hpp"
00012
00013 namespace lava {
00014
00018 struct benchmark_data {
00020     ms time = ms{10000};
00021
00023     ms offset = ms{5000};
00024
00026     string file = _benchmark_json_;
00027
00029     string path;
00030
00032     bool exit = true;
00033
00035     ui32 buffer_size = 100000;
00036
00038     using list = std::vector<ui32>;
00039
00041     list values;
00042
00044     index current = 0;
00045
00047     ms start_timestamp = ms{0};
00048 };
00049
00056 bool parse_benchmark(cmd_line cmd_line, benchmark_data& data);
00057
00063 void benchmark(frame& app, benchmark_data& data);
00064
00070 bool write_frames_json(benchmark_data& data);
00071
00072 } // namespace lava

```

## 5.7 liblava/app/camera.hpp File Reference

First Person / Look At camera.

```

#include "liblava/frame/gamepad.hpp"
#include "liblava/frame/input.hpp"
#include "liblava/resource/buffer.hpp"

```

## Classes

- struct [lava::camera](#)

*First Person / Look At camera.*

## 5.7.1 Detailed Description

First Person / Look At camera.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.8 camera.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/frame/gamepad.hpp"
00011 #include "liblava/frame/input.hpp"
00012 #include "liblava/resource/buffer.hpp"
00013
00014 namespace lava {
00015
00019 struct camera : entity {
00021     using ptr = camera*;
00022
00024     using s_ptr = std::shared_ptr<camera>;
00025
00027     using s_map = std::map<id, s_ptr>;
00028
00030     using s_list = std::vector<s_ptr>;
00031
00035     enum class mode : index {
00036         first_person = 0,
00037         look_at,
00038     };
00039
00045     bool create(device::ptr device);
00046
00050     void destroy();
00051
00055     void update_projection();
00056
00062     void update_view(delta dt, mouse_position mouse_pos);
00063
00069     void update_view(delta dt, gamepad::ref pad);
00070
00075     mat4 get_view() const;
00076
00081     mat4 get_projection() const;
00082
00087     mat4 calc_view_projection() const;
00088
00094     bool handle(key_event::ref event);
00095
00102     bool handle(mouse_button_event::ref event,
00103                 mouse_position mouse_pos);
00104
00110     bool handle(scroll_event::ref event);
00111
00116     bool valid() const {
00117         return m_data ? m_data->valid() : false;
00118     }
00119
00124     VkDescriptorBufferInfo const* get_descriptor_info() const {
00125         return m_data ? m_data->get_descriptor_info() : nullptr;
00126     }
00127
00131     void upload();
00132
00136     void stop();
00137
00141     void reset();
```

```

00142
00147     void set_active(bool value = true) {
00148         m_active = value;
00149     }
00150
00155     bool activated() const {
00156         return m_active;
00157     }
00158
00163     bool moving() const {
00164         return m_move_up || m_move_down || m_move_left || m_move_right;
00165     }
00166
00174     void set_movement_keys(keys_ref up, keys_ref down,
00175                           keys_ref left, keys_ref right) {
00176         m_up_keys = up;
00177         m_down_keys = down;
00178         m_left_keys = left;
00179         m_right_keys = right;
00180     }
00181
00183     v3 position = v3(0.f);
00184
00186     v3 rotation = v3(0.f);
00187
00189     r32 rotation_speed = 20.f;
00190
00192     r32 movement_speed = 1.f;
00193
00195     r32 zoom_speed = 20.f;
00196
00198     r32 fov = 60.f;
00199
00201     r32 z_near = 0.1f;
00202
00204     r32 z_far = 256.f;
00205
00207     r32 aspect_ratio = 1.77f;
00208
00210     mode mode = mode::first_person;
00211
00213     bool lock_z = false;
00214
00216     bool lock_rotation = false;
00217
00218 private:
00223     void move_first_person(delta dt);
00224
00226     bool m_active = true;
00227
00229     bool m_move_up = false;
00230
00232     bool m_move_down = false;
00233
00235     bool m_move_left = false;
00236
00238     bool m_move_right = false;
00239
00241     bool m_rotate = false;
00242
00244     bool m_translate = false;
00245
00247     r64 m_mouse_pos_x = 0.0;
00248
00250     r64 m_mouse_pos_y = 0.0;
00251
00253     r64 m_scroll_pos = 0.0;
00254
00256     keys m_up_keys{key::w};
00257
00259     keys m_down_keys{key::s};
00260
00262     keys m_left_keys{key::a};
00263
00265     keys m_right_keys{key::d};
00266
00268     buffer::s_ptr m_data;
00269
00271     size_t m_size = sizeof(mat4) * 2;
00272
00274     mat4 m_projection = mat4(0.f);
00275
00277     mat4 m_view = mat4(0.f);
00278 };
00279
00280 } // namespace lava

```

## 5.9 liblava/app/config.hpp File Reference

Application configuration.

```
#include "liblava/app/imgui.hpp"
#include "liblava/frame/window.hpp"
#include "liblava/fwd.hpp"
#include "liblava/resource/format.hpp"
```

### Classes

- struct [lava::app\\_config](#)  
*Application configuration.*

### Functions

- void [lava::set\\_window\\_icon](#) ([window](#) &[window](#), [string\\_ref](#) icon\_file="icon.png")  
*Set the window icon.*

### 5.9.1 Detailed Description

Application configuration.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.9.2 Function Documentation

#### 5.9.2.1 set\_window\_icon()

```
void lava::set_window_icon (
    window & window,
    string\_ref icon_file = "icon.png")
```

Set the window icon.

#### Parameters

<i>window</i>	Target window
<i>icon_file</i>	Icon file

## 5.10 config.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/app/imgui.hpp"
00011 #include "liblava/frame/window.hpp"
00012 #include "liblava/fwd.hpp"
00013 #include "liblava/resource/format.hpp"
00014
00015 namespace lava {
00016
00020 struct app_config : configurable {
00022     app* context = nullptr;
00023
00025     name org = _liblava_;
00026
00028     name ext = "zip";
00029
00031     bool save_window = true;
00032
00034     bool handle_key_events = true;
00035
00037     bool v_sync = false;
00038
00040     bool triple_buffer = true;
00041
00043     ui32 fps_cap = 0;
00044
00046     surface_format_request surface;
00047
00049     index physical_device = 0;
00050
00052     ImGui::font imgui_font;
00053
00055     string name_id = _default_;
00056
00058     void set_json(json_ref j) override;
00059
00061     json get_json() const override;
00062
00066     void update_window_state();
00067
00069     window::state::optional window_state;
00070 };
00071
00077 void set_window_icon(window& window, string_ref icon_file = "icon.png");
00078
00079 } // namespace lava

```

## 5.11 liblava/app/forward\_shading.hpp File Reference

Forward shading.

```

#include "liblava/block/render_pass.hpp"
#include "liblava/frame/render_target.hpp"

```

### Classes

- struct [lava::forward\\_shading](#)  
*Forward shading.*

### 5.11.1 Detailed Description

Forward shading.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.12 forward\_shading.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/render_pass.hpp"
00011 #include "liblava/frame/render_target.hpp"
00012
00013 namespace lava {
00014
00018 struct forward_shading {
00022     explicit forward_shading() = default;
00023
00027     ~forward_shading() {
00028         destroy();
00029     }
00030
00036     bool create(render_target::s_ptr target);
00037
00041     void destroy();
00042
00047     render_pass::s_ptr get_pass() const {
00048         return m_pass;
00049     }
00050
00055     VkRenderPass get_vk_pass() const {
00056         return m_pass->get();
00057     }
00058
00063     image::s_ptr get_depth_stencil() const {
00064         return m_depth_stencil;
00065     }
00066
00067 private:
00069     render_target::s_ptr m_target;
00070
00072     render_pass::s_ptr m_pass;
00073
00075     image::s_ptr m_depth_stencil;
00076 };
00077
00078 } // namespace lava

```

## 5.13 liblava/app/icon.hpp File Reference

App default icon data.

```
#include "liblava/core/types.hpp"
```

## Variables

- constexpr `uchar` `lava::icon_png` []  
*App default icon data Generate: xxd -i res/icon.png.*
- constexpr `ui32` `lava::icon_png_len` = 13773  
*App default icon data length.*

## 5.13.1 Detailed Description

App default icon data.

## Authors

Lava Block OÜ and contributors

## Copyright

Copyright (c) 2018-present, MIT License

## 5.14 icon.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011
00012 namespace lava {
00013
00018     constexpr uchar icon_png[] = {
00019         0x89, 0x50, 0x4e, 0x47, 0x0d, 0x0a, 0x1a, 0x0a, 0x00, 0x00, 0x00, 0x0d,
00020         0x49, 0x48, 0x44, 0x52, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x00,
00021         0x08, 0x06, 0x00, 0x00, 0x00, 0x5c, 0x72, 0xa8, 0x66, 0x00, 0x00, 0x00,
00022         0x06, 0x62, 0x4b, 0x47, 0x44, 0x00, 0xff, 0x00, 0xff, 0x00, 0xff, 0xa0,
00023         0xbd, 0xa7, 0x93, 0x00, 0x00, 0x00, 0x09, 0x70, 0x48, 0x59, 0x73, 0x00,
00024         0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x01, 0x01, 0x38, 0x22, 0xf4, 0x40,
00025         0x00, 0x00, 0x00, 0x07, 0x74, 0x49, 0x4d, 0x45, 0x07, 0xe6, 0x07, 0x17,
00026         0x0a, 0x01, 0x20, 0x5c, 0xc5, 0x75, 0x56, 0x00, 0x00, 0x20, 0x00, 0x49,
00027         0x44, 0x41, 0x54, 0x78, 0xda, 0xed, 0x9d, 0x79, 0x7c, 0xcc, 0xd7, 0xf7,
00028         0xff, 0x9f, 0xd9, 0xf7, 0x5d, 0x42, 0x82, 0x88, 0x9d, 0x24, 0x76, 0xaa,
00029         0x14, 0x45, 0x29, 0x2d, 0x45, 0xab, 0x25, 0x76, 0xad, 0xda, 0x5b, 0x54,
00030         0xed, 0xb5, 0xd6, 0x56, 0x94, 0x0a, 0x4a, 0xed, 0x6a, 0x2b, 0xa9, 0xa5,
00031         0xd6, 0xaa, 0xda, 0x77, 0x8a, 0x24, 0x62, 0xc9, 0x82, 0x48, 0x44, 0x16,
00032         0xd9, 0xd7, 0xc9, 0x7e, 0x7f, 0x7f, 0xa4, 0xbe, 0xbf, 0xcf, 0xe7, 0xd3,
00033         0xcc, 0x7b, 0x46, 0x11, 0x93, 0xcc, 0x7d, 0x3e, 0x1e, 0xf3, 0x0f, 0x67,
00034         0x26, 0x33, 0xf7, 0x9e, 0xf3, 0x7a, 0xdf, 0xe5, 0x9c, 0x7b, 0x41, 0x22,
00035         0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89,
00036         0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24,
00037         0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91,
00038         0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89, 0x44,
00039         0xa2, 0x9b, 0x38, 0x01, 0xab, 0x80, 0x1b, 0x40, 0x07, 0xd9, 0x1c, 0x12,
00040         0x89, 0x7e, 0x60, 0x0a, 0x8c, 0x05, 0x12, 0x01, 0xf1, 0x1f, 0xaf, 0x7d,
00041         0x40, 0x0d, 0xd9, 0x3c, 0x12, 0x49, 0xd9, 0xe5, 0x5d, 0xe0, 0xce, 0xff,
00042         0x04, 0xfe, 0x7f, 0xbe, 0x54, 0xc0, 0x02, 0xc0, 0x56, 0x36, 0x95, 0x44,
00043         0x52, 0x76, 0xa8, 0x09, 0x1c, 0x50, 0x08, 0xfc, 0xff, 0x7d, 0xc5, 0x01,
00044         0x83, 0x01, 0x43, 0xd9, 0x74, 0x12, 0x49, 0xe9, 0xc5, 0x0e, 0x58, 0x02,
00045         0xe4, 0x3c, 0x47, 0xf0, 0xff, 0xe7, 0xeb, 0x2a, 0xd0, 0x52, 0x36, 0xa3,
00046         0x44, 0x52, 0xba, 0x30, 0x02, 0x86, 0x00, 0x31, 0xff, 0x32, 0xf0, 0xff,
00047         0xf3, 0x55, 0x08, 0x6c, 0x05, 0x2a, 0xca, 0x66, 0x95, 0x48, 0x74, 0x9f,
00048         0xd6, 0xc0, 0x75, 0x4d, 0x81, 0xed, 0xea, 0xea, 0x2a, 0x76, 0xee, 0xdc,
00049         0x29, 0xae, 0x5d, 0xbb, 0x26, 0x7a, 0xf9, 0xf4, 0xd2, 0x46, 0x08, 0x32,
00050         0x81, 0x19, 0x80, 0xa5, 0x6c, 0x62, 0x89, 0x44, 0xf7, 0x70, 0x07, 0x76,
00051         0xfc, 0xfd, 0xc4, 0x56, 0x0c, 0xe6, 0x39, 0x73, 0xe6, 0x88, 0x27, 0x4f,
00052         0x9e, 0x88, 0x67, 0xe4, 0xe4, 0xe4, 0x88, 0x63, 0xc7, 0x8e, 0x89, 0xa6,
00053         0xcd, 0x9a, 0x6a, 0x23, 0x04, 0xe1, 0x40, 0x2f, 0xd9, 0xdc, 0x12, 0x89,
00054         0x6e, 0x60, 0x09, 0xcc, 0x06, 0xb2, 0x34, 0x05, 0xef, 0xc0, 0x41, 0x03,
00055         0x45, 0x40, 0x40, 0x80, 0x28, 0x2c, 0x2c, 0x14, 0xc5, 0x91, 0x92, 0x92,
00056         0x22, 0xd6, 0xad, 0x5b, 0x27, 0x8c, 0x8c, 0x8c, 0xb4, 0x11, 0x82, 0xd3,
```

```
00057 0x40, 0x03, 0xd9, 0xfc, 0x12, 0xc9, 0xeb, 0xc1, 0x00, 0xf0, 0x01, 0x22,
00058 0x35, 0x05, 0xeb, 0x1b, 0x6f, 0x36, 0x17, 0xc7, 0x8f, 0x1f, 0x17, 0x39,
00059 0x39, 0x39, 0x42, 0x1b, 0x1e, 0x3e, 0x7c, 0x28, 0xc6, 0x7f, 0x3d, 0x5e,
00060 0x1b, 0x11, 0x28, 0x00, 0xd6, 0x03, 0x2e, 0xb2, 0x3b, 0x24, 0x92, 0x92,
00061 0xa3, 0x31, 0x70, 0x4e, 0x53, 0x80, 0x5a, 0xda, 0x5b, 0x8a, 0xf5, 0xeb,
00062 0xd7, 0x8b, 0xd4, 0xd4, 0x54, 0xf1, 0xbc, 0x14, 0x14, 0x14, 0x88, 0xab,
00063 0x57, 0xaf, 0x8a, 0xee, 0x1f, 0x76, 0xd7, 0x46, 0x08, 0x52, 0x80, 0xaf,
00064 0x29, 0x4a, 0x32, 0x92, 0x48, 0x24, 0xaf, 0x88, 0x0a, 0xc0, 0x06, 0x6d,
00065 0xe6, 0xf9, 0x93, 0x26, 0x4d, 0x12, 0xe1, 0xe1, 0xe1, 0xe2, 0x45, 0x51,
00066 0xa9, 0x54, 0xe2, 0xc0, 0x81, 0x03, 0xa2, 0x6e, 0xdd, 0xba, 0xda, 0x08,
00067 0x41, 0x08, 0xd0, 0x59, 0x76, 0x93, 0x44, 0xf2, 0x72, 0x31, 0x03, 0x26,
00068 0x01, 0x69, 0x9a, 0x82, 0xf0, 0x93, 0x5e, 0x9f, 0x88, 0x6b, 0xd7, 0xae,
00069 0x89, 0x82, 0xc2, 0x02, 0xf1, 0x32, 0x89, 0x8f, 0x8f, 0x17, 0xcb, 0x96,
00070 0x2d, 0x13, 0x06, 0x06, 0x06, 0xda, 0x08, 0xc1, 0x51, 0xc0, 0x53, 0x76,
00071 0x9b, 0x44, 0xf2, 0xe2, 0x74, 0x01, 0x42, 0x35, 0x05, 0x9d, 0x57, 0x3d,
00072 0x2f, 0x71, 0xf0, 0xe0, 0x41, 0x91, 0x9d, 0x9d, 0x2d, 0x5e, 0x25, 0xc1,
00073 0xc1, 0xc1, 0x62, 0xcc, 0xb8, 0x31, 0xda, 0x88, 0x40, 0x0e, 0xe0, 0x0b,
00074 0x38, 0xc8, 0x2e, 0x94, 0x48, 0x9e, 0x1f, 0x2f, 0xe0, 0x4f, 0x4d, 0x81,
00075 0x66, 0x61, 0x61, 0x21, 0x96, 0x2f, 0x5f, 0x2e, 0x12, 0x12, 0x12, 0x44,
00076 0x49, 0x91, 0x9f, 0x9f, 0x2f, 0x2e, 0x5c, 0xb8, 0x2d, 0x3a, 0x75, 0xea,
00077 0xa4, 0x8d, 0x10, 0xc4, 0x03, 0x43, 0x01, 0x63, 0xd9, 0xa5, 0x12, 0x89,
00078 0x66, 0x1c, 0x81, 0x1f, 0x81, 0x3c, 0x4d, 0xc1, 0x35, 0x6e, 0xdc, 0x38,
00079 0x11, 0x1a, 0x1a, 0x2a, 0x5e, 0x17, 0x99, 0x99, 0x99, 0x62, 0xe7, 0xce,
00080 0x9d, 0xa2, 0x72, 0xe5, 0xca, 0xda, 0x08, 0x81, 0x3f, 0xd0, 0x4e, 0x76,
00081 0xaf, 0x44, 0x52, 0x3c, 0x46, 0xc0, 0x48, 0x20, 0x41, 0x53, 0x30, 0x75,
00082 0xee, 0xdc, 0x59, 0x5c, 0xba, 0x74, 0x49, 0x14, 0x14, 0x14, 0x08, 0x5d,
00083 0x20, 0x3a, 0x3a, 0x5a, 0x2c, 0x58, 0xb0, 0x40, 0xdb, 0xd4, 0xe2, 0xbd,
00084 0x40, 0x35, 0xd9, 0xdd, 0x12, 0xc9, 0xff, 0xe7, 0x1d, 0xe0, 0xb6, 0xa6,
00085 0xe0, 0x71, 0x72, 0x72, 0x12, 0xbb, 0x76, 0xed, 0x12, 0x99, 0x99, 0x99,
00086 0x42, 0x17, 0x09, 0x0a, 0x0a, 0x12, 0x43, 0x86, 0x0c, 0xd1, 0x46, 0x04,
00087 0x54, 0xc0, 0x3c, 0xc0, 0x5a, 0x76, 0xbd, 0x44, 0x9f, 0xa9, 0x0e, 0xfc,
00088 0xa6, 0xcd, 0x93, 0x73, 0xf1, 0xe2, 0xc5, 0x22, 0x26, 0x36, 0x46, 0xe8,
00089 0x3a, 0x79, 0x79, 0x79, 0xe2, 0xc4, 0x89, 0x13, 0xa2, 0x45, 0x8b, 0x16,
00090 0xda, 0x08, 0xc1, 0x63, 0xa0, 0x1f, 0xb2, 0xec, 0x58, 0xa2, 0x67, 0x58,
00091 0x53, 0x74, 0xf0, 0x46, 0xb6, 0xa6, 0x20, 0x19, 0x3e, 0x7c, 0xb8, 0xb8,
00092 0x7b, 0xef, 0xae, 0xda, 0xf4, 0x5d, 0x5d, 0x25, 0x25, 0x35, 0x45, 0x6c,
00093 0xdc, 0xb8, 0x51, 0x54, 0x70, 0xad, 0xa0, 0x6d, 0xd9, 0xf1, 0x9b, 0xd2,
00094 0x2d, 0x24, 0xfa, 0x30, 0xcf, 0x1f, 0x0c, 0x44, 0x6b, 0x0a, 0x8a, 0x37,
00095 0xdf, 0x7c, 0x53, 0x9c, 0x3c, 0x79, 0x52, 0xe4, 0xe5, 0xe5, 0x89, 0xd2,
00096 0x4c, 0x64, 0x64, 0xa4, 0x98, 0x31, 0x63, 0xc6, 0xf3, 0x94, 0x1d, 0xbb,
00097 0x4a, 0x37, 0x91, 0x94, 0x45, 0x5a, 0x01, 0xd7, 0x34, 0x05, 0x42, 0x95,
00098 0x2a, 0x55, 0xc4, 0x96, 0x2d, 0x5b, 0x44, 0x5a, 0x5a, 0x9a, 0x28, 0x2b,
00099 0x14, 0x16, 0x16, 0x0a, 0x7f, 0x7f, 0x7f, 0xf1, 0xc9, 0x27, 0x9f, 0x68,
00100 0x23, 0x04, 0x69, 0xc0, 0x34, 0x8a, 0x92, 0x9f, 0x24, 0x92, 0x52, 0x8f,
00101 0x1b, 0xb0, 0x5d, 0x9b, 0x79, 0xfe, 0xec, 0xd9, 0xb3, 0x45, 0x54, 0x54,
00102 0x94, 0x28, 0xab, 0x64, 0x67, 0x67, 0x8b, 0x43, 0x87, 0x0e, 0x09, 0x2f,
00103 0x2f, 0x2f, 0x6d, 0x84, 0xe0, 0x11, 0xf0, 0xa1, 0x74, 0x1f, 0x49, 0x69,
00104 0xc5, 0x12, 0x98, 0x0e, 0x64, 0x68, 0x72, 0xf6, 0xde, 0xbd, 0x7b, 0x8b,
00105 0x80, 0x80, 0x00, 0xa1, 0x2f, 0x24, 0xa5, 0x24, 0x89, 0x1f, 0x7f, 0xfc,
00106 0x51, 0x98, 0x9a, 0x9a, 0x6a, 0x5b, 0x76, 0x5c, 0x5f, 0xba, 0x93, 0xa4,
00107 0x34, 0xe1, 0x43, 0xd1, 0xc1, 0x19, 0x8a, 0xce, 0xdd, 0xbc, 0x79, 0x73,
00108 0x71, 0xf8, 0xf0, 0x61, 0xad, 0xcb, 0x74, 0xcb, 0x1a, 0xe1, 0xe1, 0xe1,
00109 0xe2, 0xeb, 0xaf, 0xbf, 0xd6, 0x46, 0x04, 0xf2, 0x80, 0x35, 0x14, 0xdd,
00110 0x65, 0x20, 0x91, 0xe8, 0x2c, 0x8d, 0x80, 0xb3, 0x9a, 0x1c, 0xda, 0xd6,
00111 0xd6, 0x56, 0xac, 0x5d, 0xbb, 0x56, 0x24, 0x27, 0x27, 0x0b, 0x7d, 0xa7,
00112 0xa0, 0xa0, 0x40, 0x5c, 0xba, 0x74, 0x49, 0x74, 0xee, 0xdc, 0x59, 0x1b,
00113 0x21, 0x48, 0x06, 0xc6, 0x03, 0x26, 0xd2, 0xd5, 0x24, 0xba, 0x84, 0x0b,
00114 0xb0, 0x16, 0x2d, 0xd2, 0x77, 0x27, 0x4e, 0x9c, 0x28, 0x1e, 0x45, 0x3c,
00115 0x12, 0x92, 0xff, 0x26, 0x2b, 0x2b, 0x4b, 0xec, 0xd9, 0xb3, 0x47, 0xdb,
00116 0xb2, 0xeb, 0x7b, 0xc8, 0xb2, 0x63, 0x89, 0x0e, 0x60, 0x02, 0x4c, 0x04,
00117 0x92, 0x34, 0x39, 0x6d, 0xf7, 0xee, 0xdd, 0xc5, 0xe5, 0x2b, 0x57, 0x74,
00118 0x26, 0x7d, 0x57, 0x57, 0x89, 0x8f, 0x8f, 0x17, 0xcb, 0x96, 0x2f, 0xd3,
00119 0x36, 0xad, 0xf8, 0x10, 0x45, 0x77, 0x1e, 0x48, 0x24, 0x25, 0x8e, 0x56,
00120 0x65, 0xba, 0x55, 0xaa, 0x54, 0x11, 0xbf, 0xfd, 0xf6, 0x9b, 0x50, 0xa9,
00121 0x54, 0x32, 0xba, 0x9f, 0xb3, 0xec, 0x78, 0xe8, 0xd0, 0xa1, 0xda, 0x96,
00122 0x1d, 0x2f, 0x05, 0xec, 0xa5, 0x4b, 0x4a, 0x4a, 0x82, 0x3a, 0x14, 0x1d,
00123 0x78, 0xa1, 0xd1, 0x39, 0x97, 0x2f, 0x5f, 0x2e, 0x12, 0x13, 0x13, 0x65,
00124 0x34, 0xbf, 0x40, 0xd9, 0xf1, 0xe9, 0xd3, 0xa7, 0xc5, 0x3b, 0xef, 0xbc,
00125 0xa3, 0x8d, 0x10, 0xc4, 0x02, 0xc3, 0x28, 0x4a, 0xb6, 0x92, 0x48, 0x5e,
00126 0x3a, 0x76, 0xc0, 0x72, 0xb4, 0xb8, 0x65, 0x67, 0xe4, 0xc8, 0x91, 0x22,
00127 0x24, 0x24, 0x44, 0x46, 0xf0, 0x4b, 0x22, 0x3d, 0x3d, 0x5d, 0xfc, 0xf2,
00128 0xcb, 0x2f, 0xda, 0x96, 0x1d, 0xdf, 0xa4, 0xe8, 0x6e, 0x04, 0x89, 0xe4,
00129 0xa5, 0x60, 0x4c, 0x51, 0x99, 0x6e, 0xbc, 0x26, 0xe7, 0x6b, 0xdf, 0xbe,
00130 0xbd, 0x38, 0x77, 0xfe, 0xbc, 0xc8, 0xcf, 0xcf, 0x97, 0x51, 0xfb, 0x8a,
00131 0xca, 0x8e, 0xbf, 0xfd, 0xf6, 0x5b, 0x6d, 0xd7, 0x07, 0x76, 0x01, 0x1e,
00132 0xd2, 0x7d, 0x25, 0x2f, 0x42, 0x3b, 0x20, 0x40, 0x93, 0xb3, 0x55, 0xae,
00133 0x5c, 0x59, 0xec, 0xde, 0xbd, 0x5b, 0x67, 0xcb, 0x74, 0xcb, 0x12, 0x85,
00134 0x85, 0x85, 0x22, 0x30, 0x30, 0x50, 0x0c, 0x18, 0x30, 0x40, 0x1b, 0x11,
00135 0xc8, 0x00, 0xbe, 0x45, 0x96, 0x1d, 0x4b, 0x9e, 0x13, 0x0f, 0x60, 0x8f,
00136 0x36, 0x4f, 0x9a, 0x79, 0xf3, 0xe6, 0x89, 0x98, 0x18, 0x19, 0x99,
00137 0x25, 0x4c, 0x4e, 0x4e, 0x8e, 0x38, 0x71, 0xe2, 0x84, 0x78, 0xe3, 0x8d,
00138 0x37, 0xb4, 0x11, 0x82, 0x28, 0xa0, 0xaf, 0x74, 0x6b, 0x89, 0x26, 0x6c,
00139 0x80, 0xf9, 0x14, 0x1d, 0x58, 0xa1, 0x7c, 0xcb, 0xce, 0xe0, 0x81, 0xe2,
00140 0xf6, 0xed, 0xdb, 0xa5, 0xae, 0x4c, 0xb7, 0xac, 0x91, 0x96, 0x96, 0x26,
00141 0x36, 0x6c, 0xdc, 0x20, 0x8c, 0x8d, 0x8d, 0xb5, 0x11, 0x82, 0xf3, 0x40,
00142 0x53, 0xe9, 0xe6, 0x92, 0xe2, 0x18, 0xf0, 0xf7, 0x93, 0x42, 0xd1, 0x89,
00143 0x5a, 0xb7, 0x6e, 0x5d, 0x26, 0xca, 0x74, 0xcb, 0x1a, 0x11, 0x11, 0x11,
```



```
00144 0x62, 0xf2, 0xe4, 0xc9, 0xda, 0xa6, 0x15, 0x6f, 0x46, 0x96, 0x1d, 0x4b,
00145 0xfe, 0xa6, 0x25, 0x70, 0x49, 0x93, 0xe3, 0x98, 0x98, 0x88, 0x4d,
00146 0x9b, 0x37, 0x95, 0xa9, 0x32, 0xdd, 0xb2, 0x98, 0x56, 0x7c, 0xfd, 0xc6,
00147 0x75, 0xd1, 0xb3, 0x67, 0x4f, 0x6d, 0xcb, 0x8e, 0xa7, 0x22, 0xcb, 0x8e,
00148 0xf5, 0x96, 0x4a, 0x14, 0x1d, 0x40, 0x51, 0xa0, 0xc9, 0x59, 0xbe, 0x99,
00149 0xfe, 0x8d, 0x88, 0x8c, 0x8c, 0x94, 0x11, 0x56, 0x8a, 0xd6, 0x07, 0x0e,
00150 0x1f, 0x3e, 0xac, 0x6d, 0xd9, 0x71, 0x18, 0xd0, 0x4d, 0x86, 0x83, 0xfe,
00151 0x60, 0x0e, 0x7c, 0x83, 0x16, 0xb7, 0xec, 0xf4, 0xed, 0xd7, 0x57, 0xf8,
00152 0xfb, 0xfb, 0xcb, 0xf4, 0xdd, 0x52, 0x4a, 0x62, 0x62, 0xa2, 0xf0, 0x5d,
00153 0xe1, 0x2b, 0x6c, 0x6d, 0x6d, 0xb5, 0x11, 0x82, 0xe3, 0x40, 0x3d, 0x19,
00154 0x1e, 0x65, 0x17, 0x03, 0xa0, 0x27, 0x70, 0x5f, 0x93, 0x33, 0x78, 0x79,
00155 0x79, 0x89, 0xa3, 0x47, 0x8f, 0xea, 0x6d, 0x99, 0x6e, 0x59, 0x23, 0x2c,
00156 0x2c, 0x4c, 0x8c, 0x1d, 0x37, 0x56, 0xdb, 0xf5, 0x81, 0xd5, 0xc8, 0xb2,
00157 0xe3, 0x32, 0x47, 0x43, 0xe0, 0xa4, 0x36, 0xdb, 0x7a, 0xab, 0x57, 0xaf,
00158 0x16, 0x49, 0x49, 0x49, 0x32, 0x6a, 0xca, 0xe0, 0xfa, 0xc0, 0xe5, 0xcb,
00159 0x97, 0xc5, 0xfb, 0xef, 0xbf, 0xaf, 0x8d, 0x10, 0x24, 0x02, 0xa3, 0x90,
00160 0xb7, 0x19, 0x95, 0x7a, 0xca, 0x51, 0x54, 0xa6, 0x9b, 0xaf, 0xa9, 0xd3,
00161 0x27, 0x4c, 0x98, 0x20, 0x1e, 0x3c, 0x78, 0xa0, 0x77, 0x81, 0xf1, 0xaa,
00162 0xef, 0x12, 0xd4, 0x35, 0xb2, 0xb2, 0xb2, 0x84, 0x9f, 0x9f, 0x9f, 0x70,
00163 0x75, 0x75, 0xd5, 0x46, 0x08, 0x6e, 0x03, 0xef, 0xca, 0x30, 0x2a, 0x7d,
00164 0x98, 0x02, 0x5f, 0xa2, 0x45, 0x99, 0x6e, 0x97, 0xae, 0x5d, 0xc4, 0xd5,
00165 0xab, 0x57, 0xf5, 0x6e, 0x9e, 0x1f, 0x1b, 0x1b, 0x2b, 0x16, 0x2c, 0x58,
00166 0x20, 0x3a, 0xbd, 0xdb, 0x49, 0x5c, 0xbc, 0x78, 0x51, 0x2f, 0x7f, 0xff,
00167 0x92, 0xef, 0x97, 0x68, 0x9b, 0x56, 0x7c, 0x10, 0x59, 0x76, 0x5c, 0x6a,
00168 0x78, 0x0f, 0xb8, 0xab, 0xa9, 0x53, 0xab, 0x56, 0xad, 0x2a, 0xf6, 0xec,
00169 0xd9, 0x23, 0xb2, 0xb2, 0xb2, 0xf4, 0xca, 0xf1, 0x33, 0x33, 0x33, 0xc5,
00170 0xae, 0x5d, 0xbb, 0x44, 0xc5, 0x8a, 0x15, 0xff, 0xab, 0x3d, 0xbe, 0xf8,
00171 0xe2, 0x0b, 0x11, 0x16, 0x16, 0xa6, 0x57, 0x6d, 0x51, 0x58, 0x58, 0x28,
00172 0x82, 0x83, 0x83, 0xc5, 0xc8, 0x91, 0x23, 0xb5, 0x2d, 0x3b, 0x5e, 0x48,
00173 0x51, 0xb2, 0x98, 0x44, 0x07, 0xa9, 0x8d, 0x16, 0x65, 0xba, 0x06, 0x06,
00174 0x06, 0x62, 0xd9, 0x0f, 0xcb, 0x44, 0x5c, 0x5c, 0x9c, 0x5e, 0x39, 0x7b,
00175 0x7e, 0x7e, 0xbe, 0x38, 0x77, 0xee, 0x9c, 0xe8, 0xf8, 0x5e, 0x47, 0xc5,
00176 0xb6, 0xf9, 0xf1, 0xc7, 0x1f, 0xf5, 0xae, 0x84, 0x39, 0x2f, 0x2f, 0x4f,
00177 0x9c, 0x3e, 0x7d, 0x5a, 0xbc, 0xf5, 0xd6, 0x5b, 0xda, 0x08, 0x41, 0x0c,
00178 0xf0, 0xa9, 0x5c, 0x1f, 0xd0, 0x1d, 0xec, 0x80, 0x25, 0x68, 0x51, 0xa6,
00179 0x3b, 0x6a, 0xf4, 0x28, 0xbd, 0x2c, 0xd3, 0x0d, 0x0d, 0xd5, 0xf6,
00180 0x29, 0x27, 0x00, 0x51, 0xaf, 0x5e, 0x3d, 0x71, 0xf0, 0xe0, 0x41, 0xbd,
00181 0x5b, 0x1f, 0x48, 0x4b, 0x4b, 0x13, 0xdb, 0xb6, 0x6d, 0x13, 0xf6, 0x95,
00182 0x1c, 0xb5, 0x69, 0xa7, 0xeb, 0x14, 0xdd, 0xf5, 0x20, 0x79, 0x4d, 0x18,
00183 0x02, 0x9f, 0x03, 0x4f, 0x35, 0x75, 0x56, 0xbb, 0x76, 0xed, 0xc4, 0x99,
00184 0x33, 0x67, 0xf4, 0xae, 0x4c, 0x37, 0x3e, 0x3e, 0x5e, 0xac, 0x58, 0xb1,
00185 0x42, 0xeb, 0xc0, 0xff, 0xdf, 0x57, 0xb7, 0x6e, 0xdd, 0xc4, 0xb5, 0xbf,
00186 0xae, 0xe9, 0xdd, 0xfa, 0xc0, 0x93, 0x27, 0x4f, 0xc4, 0xdc, 0xb9, 0x73,
00187 0xb5, 0x6d, 0xa7, 0x9d, 0x14, 0x25, 0x95, 0x49, 0x4a, 0x90, 0xb6, 0x68,
00188 0x51, 0xa6, 0x6b, 0x67, 0x67, 0x27, 0xb6, 0xef, 0xd8, 0x2e, 0xd2, 0xd3,
00189 0xd3, 0xf5, 0xca, 0x81, 0x55, 0x2a, 0x95, 0xd8, 0xbf, 0x7f, 0xbf, 0x56,
00190 0x99, 0x70, 0xce, 0xce, 0xce, 0x1a, 0x6d, 0xa6, 0x4c, 0x99, 0xa2, 0x77,
00191 0x99, 0x90, 0xcf, 0xca, 0x8e, 0xfb, 0xf6, 0xed, 0xab, 0x8d, 0x08, 0x64,
00192 0x02, 0x33, 0x01, 0x2b, 0x19, 0x9a, 0xaf, 0x16, 0x77, 0x8a, 0x0e, 0x7a,
00193 0xd0, 0xd8, 0x29, 0x0b, 0xbf, 0x5b, 0xa8, 0x77, 0x65, 0xba, 0x05, 0x05,
00194 0x05, 0xe2, 0xea, 0xd5, 0xab, 0xa2, 0x4b, 0x97, 0x2e, 0x1a, 0xdb, 0xa7,
00195 0x7d, 0x15, 0x0f, 0xb1, 0xb3, 0x61, 0x63, 0x71, 0xa2, 0xba, 0x97, 0x18,
00196 0xef, 0x55, 0x47, 0x18, 0x1a, 0x19, 0x29, 0xda, 0x5b, 0x5a, 0x5a, 0x8a,
00197 0x75, 0xeb, 0xd7, 0x89, 0xd4, 0xd4, 0x54, 0xbd, 0x6a, 0xd3, 0x9c, 0x9c,
00198 0x1c, 0xf1, 0xfb, 0xef, 0xbf, 0x8b, 0x66, 0x6f, 0x35, 0xd3, 0x46, 0x08,
00199 0x22, 0x28, 0xba, 0x13, 0xc2, 0x40, 0x86, 0xea, 0xcb, 0xc5, 0x0a, 0x98,
00200 0x0d, 0x64, 0x69, 0xea, 0x84, 0xfe, 0xfd, 0xfb, 0x8b, 0x5b, 0xb7, 0x6e,
00201 0xe9, 0x5d, 0x99, 0x6e, 0xf8, 0xa3, 0x70, 0xad, 0xaa, 0xe1, 0xdc, 0xad,
00202 0x6d, 0xc4, 0x0f, 0xf5, 0xea, 0x89, 0x6b, 0x55, 0x3d, 0x85, 0xbf, 0xbd,
00203 0xc7, 0xff, 0xbd, 0x0e, 0x36, 0x68, 0x2a, 0xfa, 0x78, 0xd6, 0xd6, 0xf8,
00204 0xfe, 0x46, 0x8d, 0x1a, 0x89, 0x63, 0x7f, 0x1c, 0xd3, 0xbb, 0x2c, 0xc9,
00205 0xa4, 0xa4, 0x24, 0xb1, 0x71, 0xe3, 0x46, 0xe1, 0xe4, 0xe4, 0xa4, 0x8d,
00206 0x10, 0x9c, 0x03, 0x1a, 0xcb, 0xb0, 0x7d, 0x71, 0x0c, 0x80, 0xfe, 0xc0,
00207 0x13, 0x4d, 0x8d, 0xde, 0xa2, 0x45, 0x0b, 0x71, 0xec, 0xd8, 0x31, 0x91,
00208 0x93, 0xab, 0x5f, 0x8e, 0x99, 0x9c, 0x9c, 0x2c, 0xd6, 0xaf, 0x5f, 0xaf,
00209 0xd1, 0x29, 0xcd, 0x2d, 0x2c, 0xc4, 0x64, 0x2f, 0x6f, 0x71, 0xba, 0xba,
00210 0xf7, 0x7f, 0x05, 0xfe, 0x7f, 0xbe, 0x6e, 0x54, 0xac, 0x25, 0x36, 0x37,
00211 0x6c, 0x2a, 0x9a, 0x96, 0xd7, 0x7c, 0xa5, 0x77, 0xdf, 0xbe, 0x7d, 0x45,
00212 0x50, 0x50, 0x90, 0xde, 0x2d, 0xa8, 0x3e, 0x8a, 0x78, 0x2d, 0xa6, 0x4c,
00213 0x99, 0xa2, 0x8d, 0x08, 0xe4, 0x03, 0xeb, 0x81, 0xf2, 0xba, 0x1e, 0x60,
00214 0xba, 0x4a, 0x53, 0xc0, 0x97, 0xa2, 0x72, 0x5d, 0xb5, 0xd8, 0x3a, 0xdb,
00215 0xb1, 0x62, 0x89, 0x2f, 0x3d, 0x3e, 0xec, 0x81, 0x9d, 0xad, 0x9d, 0xde,
00216 0x28, 0x63, 0x6e, 0x6e, 0x2e, 0x27, 0x4e, 0x9e, 0x60, 0xda, 0xd4, 0x69,
00217 0x04, 0x06, 0x06, 0x2a, 0xda, 0xf6, 0xad, 0x5b, 0x87, 0x3e, 0xc6, 0x96,
00218 0x54, 0x7a, 0x9c, 0xa4, 0xd5, 0x67, 0xab, 0x6c, 0xcc, 0x38, 0x67, 0x6b,
00219 0xca, 0xf4, 0xd0, 0x60, 0xf2, 0xf3, 0xf2, 0x14, 0x6d, 0xbf, 0x9d, 0x37,
00220 0x97, 0xcf, 0x3f, 0x1b, 0x82, 0xab, 0xab, 0xfe, 0x94, 0xd7, 0x17, 0x16,
00221 0x16, 0x72, 0xfd, 0xfa, 0x75, 0x16, 0x2d, 0x5e, 0xc4, 0xbe, 0xbd, 0xfb,
00222 0x34, 0x99, 0xa7, 0x02, 0x0b, 0x28, 0x3a, 0x50, 0x36, 0x57, 0x0a, 0x80,
00223 0x66, 0xca, 0x03, 0x8b, 0x28, 0x3a, 0xa0, 0xc3, 0x50, 0xc9, 0x70, 0xea,
00224 0xd4, 0xa9, 0x8c, 0x1c, 0x39, 0x92, 0xca, 0x95, 0x2b, 0xeb, 0x8d, 0xf3,
00225 0x09, 0x21, 0x08, 0xba, 0x1d, 0xc4, 0xa2, 0xef, 0x16, 0xb1, 0x73, 0xe7,
00226 0x4e, 0x45, 0xdb, 0x37, 0xdd, 0xdd, 0x18, 0x6e, 0xeb, 0x42, 0xfd, 0x94,
00227 0x6c, 0x0c, 0x33, 0xb2, 0x9f, 0xfb, 0x6f, 0x25, 0x56, 0x74, 0xe4, 0xa0,
00228 0x71, 0x2e, 0x2b, 0x02, 0x6f, 0x2b, 0xda, 0x55, 0xae, 0x5c, 0x99, 0xf9,
00229 0x0b, 0x16, 0xd0, 0xa3, 0x7b, 0x77, 0x6c, 0x6c, 0xf4, 0x27, 0x4f, 0x26,
00230 0x3b, 0x3b, 0x9b, 0x3f, 0xfe, 0xf8, 0x83, 0x49, 0x93, 0x26, 0x11, 0x1a,
```

```
00231 0x1a, 0xaa, 0xc9, 0xfc, 0x3e, 0x45, 0xd7, 0x9a, 0x1d, 0x92, 0x02, 0x50,
00232 0x3c, 0xe6, 0xc0, 0x58, 0x8a, 0x4a, 0x75, 0x15, 0xbd, 0xe8, 0xa3, 0x9e,
00233 0x1f, 0x31, 0x75, 0xea, 0x34, 0x1a, 0x37, 0x6a, 0x84, 0xa1, 0xa1, 0xa1,
00234 0xde, 0x38, 0x5c, 0x74, 0x74, 0x34, 0xeb, 0xd7, 0xaf, 0x67, 0xf6, 0xec,
00235 0xd9, 0x8a, 0x76, 0x0e, 0xd6, 0xd6, 0x4c, 0xf6, 0x70, 0xe7, 0xed, 0x74,
00236 0x81, 0x79, 0xaa, 0xea, 0xc5, 0x04, 0x07, 0x08, 0xf7, 0x28, 0xc7, 0xcf,
00237 0x19, 0x49, 0x1c, 0xbc, 0xff, 0x50, 0xd1, 0xb6, 0x4d, 0x9b, 0x36, 0x7c,
00238 0xfb, 0xed, 0xb7, 0xb4, 0x6a, 0xd5, 0x0a, 0x23, 0x23, 0xfd, 0x39, 0x9e,
00239 0x3f, 0x29, 0x29, 0x89, 0x9d, 0x3b, 0x77, 0xf2, 0xe5, 0x97, 0x5f, 0x6a,
00240 0x63, 0xfe, 0x07, 0xf0, 0x15, 0x45, 0xd7, 0x9b, 0x49, 0x01, 0xf8, 0x9b,
00241 0xee, 0xc0, 0x32, 0xa0, 0x9a, 0x92, 0x51, 0xa3, 0xc6, 0x8d, 0xf8, 0x76,
00242 0xce, 0xb7, 0x74, 0xec, 0xd8, 0x11, 0x33, 0x33, 0xfd, 0x39, 0xc8, 0x25,
00243 0x3d, 0x3d, 0x9d, 0x3d, 0x7b, 0xf6, 0xf0, 0xe5, 0x97, 0x5f, 0x92, 0x99,
00244 0x99, 0xa9, 0x68, 0x3b, 0xde, 0xbb, 0x1e, 0x5d, 0x85, 0x31, 0x0e, 0x4f,
00245 0x92, 0x5f, 0xea, 0x77, 0xc8, 0xb7, 0x36, 0xe3, 0x86, 0xb5, 0x09, 0xbe,
00246 0x4f, 0xa3, 0xb8, 0x97, 0x94, 0xa2, 0x68, 0x3b, 0x62, 0xc4, 0x08, 0xc6,
00247 0x8e, 0x1b, 0x4b, 0x9d, 0xda, 0x75, 0xf4, 0x6a, 0xc1, 0x2a, 0x2c, 0x2c,
00248 0x8c, 0x15, 0x2b, 0x56, 0xb0, 0x6a, 0xd5, 0x2a, 0x4d, 0xa6, 0x79, 0xc0,
00249 0x4f, 0xc0, 0x2c, 0x8a, 0x2e, 0x3c, 0xd5, 0x5b, 0x01, 0xf0, 0x06, 0x7e,
00250 0x00, 0x3a, 0x68, 0x32, 0x5c, 0xb5, 0x6a, 0x15, 0x7d, 0xfb, 0xf6, 0xc5,
00251 0xc1, 0xc1, 0x41, 0x6f, 0x1c, 0xaa, 0xa0, 0xa0, 0x80, 0x73, 0xe7, 0xcf,
00252 0x31, 0x63, 0xfa, 0x0c, 0x2e, 0x5e, 0xbc, 0xa8, 0x68, 0xdb, 0xa3, 0x7a,
00253 0x75, 0x06, 0xd9, 0x3a, 0xe2, 0x11, 0x1e, 0xff, 0x4a, 0xbf, 0x53, 0x86,
00254 0x9d, 0x39, 0xa7, 0x9d, 0x2c, 0x58, 0x13, 0x13, 0x4b, 0x4c, 0x4c, 0x8c,
00255 0xa2, 0xed, 0xf7, 0xcb, 0x96, 0x32, 0xa0, 0x5f, 0x7f, 0x5c, 0x5c, 0x5c,
00256 0xf4, 0xaa, 0xcf, 0x2e, 0x5d, 0xba, 0xc4, 0xbc, 0x79, 0xf3, 0x38, 0x7e,
00257 0xfc, 0xb8, 0xc6, 0x59, 0x16, 0x45, 0xf9, 0x03, 0xeb, 0xff, 0x16, 0x05,
00258 0xbd, 0x11, 0x00, 0x27, 0x60, 0x2e, 0x45, 0x99, 0x7c, 0x8a, 0x57, 0x3d,
00259 0x8f, 0x1b, 0x37, 0x8e, 0xd1, 0xa3, 0x47, 0x53, 0xa3, 0x46, 0x0d, 0xbd,
00260 0x7a, 0x9a, 0x04, 0x87, 0x04, 0xb3, 0x64, 0xf1, 0x12, 0x36, 0x6d, 0xda,
00261 0xa4, 0x68, 0x57, 0xdf, 0xd5, 0x99, 0xd1, 0x36, 0x2e, 0x34, 0x4e, 0xcf,
00262 0xc5, 0x58, 0x55, 0x72, 0x3e, 0x14, 0xe7, 0x6a, 0xc7, 0x3e, 0xa3, 0x02,
00263 0xd6, 0xdd, 0x56, 0x5e, 0x1f, 0xa8, 0x5a, 0xb5, 0x2a, 0x8b, 0x16, 0x2d,
00264 0xa2, 0x6b, 0xd7, 0xae, 0x58, 0x58, 0x58, 0xe8, 0x4d, 0xff, 0xa9, 0x54,
00265 0x2a, 0x0e, 0x1f, 0x3e, 0xcc, 0xe4, 0xc9, 0x93, 0x09, 0x0f, 0x0f, 0xd7,
00266 0x64, 0x1e, 0x04, 0x8c, 0x03, 0x4e, 0xe9, 0x83, 0x00, 0x78, 0x02, 0x17,
00267 0x00, 0xc5, 0x47, 0x79, 0xb7, 0x6e, 0xdd, 0x98, 0x3c, 0x79, 0x32, 0xcd,
00268 0x9b, 0x37, 0xd7, 0xab, 0xf9, 0x64, 0x6c, 0x6c, 0x2c, 0x5b, 0xb6, 0x6c,
00269 0x61, 0xea, 0xd4, 0xa9, 0x8a, 0x76, 0x26, 0xe6, 0x26, 0xcc, 0xa9, 0x53,
00270 0x8f, 0xb7, 0xd3, 0x73, 0xb0, 0x4c, 0xcc, 0x7c, 0x2d, 0xdf, 0x55, 0x58,
00271 0x9a, 0x12, 0x6c, 0x63, 0xca, 0xa6, 0x8c, 0x64, 0x4e, 0x3c, 0x79, 0xa2,
00272 0x68, 0xdb, 0xb5, 0x6b, 0x57, 0xa6, 0x4f, 0x9f, 0x4e, 0xd3, 0xa6, 0x4d,
00273 0xf5, 0xaa, 0x3f, 0xe3, 0xe2, 0xe2, 0xd8, 0xb4, 0x69, 0x13, 0xd3, 0xa7,
00274 0x4f, 0xa7, 0xb0, 0xb0, 0x50, 0x93, 0xf9, 0x0a, 0x8a, 0xd6, 0xc1, 0xca,
00275 0xb4, 0x00, 0xb4, 0x53, 0x52, 0x3a, 0x77, 0x77, 0x96, 0x2e, 0x5d,
00276 0x4a, 0xd7, 0xae, 0x5d, 0x31, 0x37, 0x37, 0xd7, 0xab, 0x27, 0xc6, 0xa1,
00277 0x43, 0x87, 0x98, 0x30, 0x61, 0x02, 0x8f, 0x1f, 0x3f, 0x56, 0x9e, 0x63,
00278 0xd7, 0xf5, 0xe4, 0x23, 0x53, 0x2b, 0x9c, 0x23, 0xe2, 0x75, 0xe2, 0xb,
00279 0xe7, 0xd9, 0x98, 0x71, 0xc5, 0xce, 0x9c, 0x1f, 0x13, 0xe3, 0x09, 0x89,
00280 0x89, 0x56, 0xb4, 0x9d, 0x30, 0x61, 0x02, 0x23, 0x46, 0x8c, 0xa0, 0x7a,
00281 0xf5, 0xea, 0x7a, 0x35, 0xa2, 0xbb, 0x73, 0xe7, 0x0e, 0x4b, 0x96, 0x2c,
00282 0xe1, 0xe7, 0x9f, 0x7f, 0x56, 0x32, 0x3b, 0x0f, 0xb4, 0x29, 0xc9, 0xef,
00283 0xa5, 0x73, 0x4b, 0xe8, 0x15, 0x2a, 0x54, 0xc0, 0xce, 0xce, 0x4e, 0x6f,
00284 0x56, 0xf7, 0x0b, 0x0a, 0x0a, 0xb8, 0x74, 0xe5, 0x12, 0x3d, 0x7b, 0xf6,
00285 0xa4, 0x77, 0xef, 0xde, 0x8a, 0xc1, 0xdf, 0xa1, 0xa2, 0x1b, 0xbb, 0xea,
00286 0xd4, 0x67, 0x58, 0x7c, 0xb6, 0xce, 0x04, 0x3f, 0x80, 0x49, 0x7a, 0x0e,
00287 0xad, 0xa3, 0x52, 0x59, 0x67, 0x6e, 0xcf, 0xf4, 0x86, 0x8d, 0x15, 0x9f,
00288 0xf0, 0xdf, 0x7f, 0xff, 0x3d, 0x35, 0xb, 0xd6, 0x64, 0xe5, 0xca, 0x95,
00289 0x24, 0x27, 0x27, 0xeb, 0x8d, 0x00, 0x78, 0x79, 0x79, 0xb1, 0x61, 0xe3,
00290 0x06, 0x36, 0x6a, 0x98, 0xd2, 0xe9, 0xc3, 0x14, 0xa0, 0x9d, 0x36, 0x73,
00291 0x1d, 0x9f, 0x3e, 0x3e, 0x4c, 0x99, 0x3c, 0x85, 0xfa, 0xf5, 0xeb, 0x63,
00292 0x60, 0x50, 0x36, 0x53, 0xab, 0x43, 0x43, 0x43, 0x59, 0xb3, 0x66, 0x0d,
00293 0xcb, 0x97, 0x2f, 0x57, 0x76, 0x9e, 0xea, 0x6e, 0x8c, 0xb2, 0x74, 0xa1,
00294 0x59, 0xb2, 0x0a, 0x93, 0x8c, 0x1c, 0x9d, 0xff, 0x5d, 0x4f, 0xdc, 0xec,
00295 0xd9, 0x55, 0x98, 0xc5, 0xf6, 0x60, 0xe5, 0xbd, 0xf1, 0xfa, 0xf5, 0xeb,
00296 0xb3, 0x60, 0xc1, 0x02, 0x3a, 0x74, 0xec, 0x88, 0x99, 0xa9, 0x69, 0x99,
00297 0x16, 0x80, 0xf4, 0xf4, 0x74, 0x0e, 0x1d, 0x39, 0xc4, 0xa4, 0x09, 0x93,
00298 0x78, 0xa2, 0x7e, 0xba, 0x54, 0xe2, 0x23, 0x00, 0x9d, 0x15, 0x80, 0x67,
00299 0xcc, 0x9e, 0x3d, 0x9b, 0xa1, 0x43, 0x87, 0xe2, 0xe6, 0xe6, 0x56, 0x66,
00300 0x9c, 0x21, 0x29, 0x29, 0x89, 0xed, 0xdb, 0xb7, 0x33, 0x76, 0xac, 0xf2,
00301 0x74, 0xcf, 0xd8, 0xc4, 0x84, 0x9e, 0x9e, 0xde, 0x74, 0xc8, 0x2e, 0xc0,
00302 0x26, 0x2e, 0xad, 0x54, 0xfd, 0xc6, 0x42, 0x13, 0x23, 0x82, 0xec, 0xcd,
00303 0xd8, 0x94, 0xa7, 0xe2, 0xdc, 0x23, 0xe5, 0x45, 0xb0, 0x5e, 0xbd, 0x7a,
00304 0x31, 0x65, 0xca, 0x14, 0x1a, 0x36, 0x6c, 0x58, 0xe6, 0xc4, 0x3e, 0x2f,
00305 0x2f, 0x8f, 0x73, 0xe7, 0xce, 0x31, 0x75, 0xea, 0x54, 0xfe, 0xfa, 0xeb,
00306 0x2f, 0x4d, 0xe6, 0x52, 0x00, 0x8a, 0xc3, 0xd2, 0xd2, 0x92, 0xf5, 0x1b,
00307 0xd6, 0xd3, 0xed, 0x83, 0x6e, 0x58, 0x5b, 0x97, 0xde, 0x8b, 0x5e, 0x73,
00308 0x72, 0x72, 0x38, 0xfe, 0xe7, 0x71, 0x26, 0x4d, 0x9c, 0x44, 0x70, 0x70,
00309 0xb0, 0xa2, 0xed, 0xc0, 0xea, 0xd5, 0xe9, 0x6d, 0x64, 0x89, 0xdb, 0xd3,
00310 0xf4, 0x52, 0x1d, 0x00, 0x39, 0x36, 0xa6, 0x5c, 0x76, 0xb6, 0x63, 0x69,
00311 0x78, 0x08, 0x51, 0xc9, 0xca, 0x22, 0x36, 0x79, 0xf2, 0x64, 0x46, 0x8d,
00312 0x1a, 0x85, 0xbb, 0xb, 0x7b, 0xa9, 0x0f, 0x7c, 0x21, 0x04, 0xfe, 0x01,
00313 0x01, 0x2c, 0xff, 0xe1, 0x07, 0xb6, 0x6d, 0xdb, 0xa6, 0xed, 0xdb, 0xf4,
00314 0x57, 0x00, 0xfa, 0xf7, 0xef, 0xcf, 0xf6, 0xed, 0xdb, 0x15, 0xdf, 0xd8,
00315 0xba, 0x75, 0x6b, 0x66, 0xcf, 0x9e, 0x4d, 0xeb, 0xd6, 0xad, 0x31, 0x31,
00316 0x31, 0x29, 0x3d, 0x4f, 0xc3, 0xc2, 0x42, 0x6e, 0xdc, 0xb8, 0xc1, 0xd2,
00317 0xa5, 0x4b, 0xd9, 0xb, 0x7b, 0xb7, 0xa2, 0xed, 0xb, 0x5e, 0x75, 0xe9,
```

```
00318    0x9f, 0x6f, 0x84, 0x57, 0x52, 0x16, 0x86, 0x79, 0x85, 0x94, 0x15, 0x52,
00319    0xdc, 0x1c, 0x38, 0x54, 0x90, 0xcd, 0xf2, 0x47, 0xa1, 0x14, 0xe6, 0x14,
00320    0xa8, 0xb5, 0xb3, 0xb2, 0xb2, 0x62, 0xe5, 0xca, 0x95, 0x7c, 0xfc, 0xf1,
00321    0xc7, 0xa5, 0x36, 0xad, 0x38, 0x32, 0x32, 0x92, 0x0d, 0x1b, 0x36, 0x30,
00322    0x77, 0xee, 0xdc, 0xe7, 0x7d, 0x6b, 0x89, 0x0b, 0xc0, 0xeb, 0xd8, 0x8f,
00323    0xa9, 0x0a, 0xc0, 0xfa, 0xdf, 0x7f, 0xbc, 0x72, 0xe5, 0x0a, 0xed, 0xda,
00324    0xb7, 0xe3, 0xfe, 0xfd, 0xfb, 0x6a, 0x17, 0xc2, 0x22, 0x23, 0x23, 0xd9,
00325    0xba, 0x75, 0x2b, 0xd1, 0xd1, 0xd1, 0xd4, 0xac, 0x59, 0x13, 0x67, 0x67,
00326    0x67, 0x9d, 0x77, 0x86, 0x88, 0x88, 0x08, 0x16, 0x2d, 0x5a, 0xc4, 0xe0,
00327    0xc1, 0x83, 0xb9, 0x73, 0xe7, 0x8e, 0x5a, 0xbb, 0xca, 0xf6, 0xf6, 0xcc,
00328    0xa9, 0x55, 0x97, 0x41, 0x19, 0x05, 0xb8, 0x25, 0x66, 0x62, 0x50, 0x28,
00329    0xca, 0xd4, 0x50, 0xd8, 0x3c, 0x3d, 0x9b, 0x06, 0x99, 0xf9, 0xbc, 0x5b,
00330    0xb1, 0x22, 0xb9, 0xb6, 0xd6, 0x04, 0xa7, 0x24, 0xab, 0x1d, 0x32, 0x1f,
00331    0x38, 0x70, 0x80, 0xf3, 0xe7, 0xcf, 0xe3, 0xee, 0xee, 0x4e, 0x65, 0xf7,
00332    0xca, 0x18, 0x19, 0x96, 0x8e, 0x6d, 0xc3, 0x94, 0x94, 0x14, 0x36, 0x6f,
00333    0xde, 0x4c, 0x97, 0x2e, 0x5d, 0x38, 0x7d, 0xfa, 0xb4, 0xa2, 0xad, 0x73,
00334    0x05, 0x67, 0xb2, 0x32, 0xb2, 0xfe, 0xe1, 0xe2, 0x14, 0x5d, 0x5c, 0xaa,
00335    0x7f, 0x23, 0x80, 0xdc, 0xdc, 0x5c, 0x4c, 0x4c, 0x4c, 0x48, 0x4f, 0x4f,
00336    0xc7, 0xef, 0x57, 0x3f, 0x86, 0x0f, 0x1b, 0x4e, 0x41, 0x41, 0x81, 0xe2,
00337    0x07, 0x2d, 0x59, 0xb2, 0x84, 0x81, 0x03, 0x07, 0xea, 0x64, 0xa6, 0x59,
00338    0x6a, 0x6a, 0x2a, 0x7b, 0xf7, 0xee, 0x65, 0xd4, 0xa8, 0x51, 0xea, 0x4,
00339    0x28, 0x2f, 0xdc, 0x4d, 0xa8, 0x53, 0x97, 0xae, 0x86, 0xa6, 0xd8, 0x45,
00340    0xa7, 0xa2, 0x0f, 0x14, 0x94, 0x33, 0xe3, 0x96, 0x95, 0x35, 0xeb, 0x53,
00341    0xe3, 0xb9, 0xfc, 0x28, 0x52, 0xd1, 0x76, 0xc0, 0x80, 0x01, 0x4c, 0x98,
00342    0x30, 0x81, 0xfa, 0xf5, 0xeb, 0xeb, 0xec, 0xef, 0xc9, 0xce, 0xce, 0xe6,
00343    0x8f, 0xe3, 0xc7, 0x99, 0x3c, 0x69, 0x12, 0x21, 0x21, 0x21, 0x8a, 0xb6,
00344    0x3e, 0x3e, 0x3e, 0x4c, 0x99, 0x32, 0x85, 0x7b, 0xf7, 0xee, 0xd1, 0xa7,
00345    0x4f, 0x9f, 0xd7, 0x3e, 0x02, 0xd0, 0xb9, 0xbd, 0x36, 0x1b, 0x1b, 0x1b,
00346    0x86, 0x7c, 0x36, 0x84, 0xf0, 0xf0, 0x70, 0x66, 0xcc, 0x9c, 0xa1, 0x68,
00347    0x3b, 0x71, 0xe2, 0x44, 0x9a, 0xbf, 0xd9, 0x1c, 0x3f, 0x3f, 0x8d,
00348    0x39, 0xf2, 0x25, 0x45, 0x6e, 0x5e, 0x2e, 0xc7, 0x8f, 0x1f, 0xa7, 0xcb,
00349    0x07, 0x5d, 0x18, 0x32, 0x64, 0x88, 0x62, 0xf0, 0xf7, 0xf1, 0xf4, 0x62,
00350    0x7f, 0x2d, 0x6f, 0xfa, 0xc5, 0xaa, 0xf4, 0x26, 0xf8, 0x01, 0x8c, 0x12,
00351    0x72, 0x68, 0x14, 0x91, 0xc8, 0x52, 0x23, 0x1b, 0x16, 0x7a, 0x37, 0xc0,
00352    0xc1, 0xc1, 0x5e, 0xad, 0xed, 0xb6, 0x6d, 0xdb, 0x68, 0xd0, 0xa0, 0x01,
00353    0x0b, 0x16, 0x2c, 0x20, 0x36, 0x36, 0x56, 0xe7, 0xa6, 0x76, 0x57, 0xae,
00354    0x5c, 0xa1, 0x77, 0xef, 0xde, 0xf4, 0xe8, 0xde, 0x5d, 0x31, 0xf8, 0x5b,
00355    0xb6, 0x6c, 0xc9, 0xf1, 0xe3, 0xc7, 0xd9, 0xba, 0x75, 0x2b, 0x0d, 0x1a,
00356    0x34, 0xd0, 0x99, 0x6d, 0x6e, 0x9d, 0x1b, 0x01, 0xfc, 0x6f, 0x03, 0xdf,
00357    0xba, 0x75, 0x8b, 0xc5, 0x8b, 0x17, 0xf3, 0xcb, 0x2f, 0xbf, 0x28, 0x7e,
00358    0x68, 0xc7, 0x4e, 0x1d, 0x99, 0x3d, 0x73, 0xf6, 0x6b, 0xcb, 0x1c, 0x14,
00359    0x42, 0x10, 0x14, 0x14, 0xc4, 0x77, 0xdf, 0x7d, 0xa7, 0xf1, 0xbb, 0xb6,
00360    0xa8, 0x54, 0x99, 0x61, 0xf6, 0xe5, 0xa8, 0x97, 0x98, 0x8e, 0x91, 0x2a,
00361    0x1f, 0x7d, 0x27, 0xde, 0xd9, 0x9a, 0xfd, 0xe4, 0xb1, 0x26, 0x4c, 0xf9,
00362    0xe9, 0xe9, 0xe2, 0xe2, 0x82, 0xaf, 0xaf, 0x2f, 0xdd, 0xba, 0x75, 0xc3,
00363    0xd2, 0xd2, 0xf2, 0xb5, 0x7e, 0xe7, 0xb0, 0xb0, 0x30, 0x7e, 0xf0, 0x5d,
00364    0xce, 0x9a, 0x1f, 0x57, 0x2b, 0xda, 0x39, 0x38, 0x38, 0xb0, 0x7c, 0xf9,
00365    0x72, 0x3e, 0xfc, 0xf0, 0xc3, 0xff, 0x5a, 0xd3, 0xf0, 0xf3, 0xf3, 0xa3,
00366    0x77, 0xef, 0xde, 0xaf, 0x7d, 0x04, 0xa0, 0xd3, 0x02, 0xf0, 0x8c, 0x9c,
00367    0x9c, 0x1c, 0x4e, 0x9c, 0x38, 0xc1, 0xf4, 0xe9, 0xd3, 0x09, 0x08, 0x08,
00368    0x50, 0xfc, 0xf0, 0x31, 0x63, 0xc7, 0xf0, 0xe5, 0x17, 0x5f, 0x96, 0x68,
00369    0xed, 0x40, 0x4c, 0x4c, 0x0c, 0x1b, 0x36, 0x6c, 0x60, 0xe6, 0xcc, 0x99,
00370    0xca, 0xf3, 0x3e, 0x47, 0x07, 0xbe, 0x2e, 0x5f, 0x91, 0xb7, 0x0b, 0x0c,
00371    0x30, 0x2f, 0xe5, 0xab, 0xfb, 0x2f, 0x5d, 0x40, 0x0d, 0x0c, 0x78, 0x50,
00372    0xd9, 0x9e, 0x9d, 0x39, 0x2a, 0xf6, 0x87, 0x28, 0xef, 0x90, 0xbc, 0xfb,
00373    0xee, 0xbb, 0x4c, 0x9b, 0x36, 0x8d, 0xb7, 0xde, 0x7a, 0x0b, 0x63, 0xe3,
00374    0x92, 0x3d, 0x9e, 0x3f, 0x26, 0x26, 0x86, 0x5f, 0x7e, 0xf9, 0x85, 0xaf,
00375    0xbf, 0xfe, 0x5a, 0x39, 0xb0, 0x0c, 0x0c, 0x98, 0x3f, 0x7f, 0x3e, 0x9f,
00376    0x7d, 0xf6, 0x19, 0xe5, 0xcb, 0xff, 0xf3, 0x50, 0x20, 0x29, 0x00, 0xcf,
00377    0x21, 0x00, 0xcf, 0x48, 0x4a, 0x4a, 0xc2, 0xcf, 0xcf, 0x8f, 0x19, 0x33,
00378    0x67, 0x90, 0x10, 0x9f, 0xa0, 0x68, 0xbb, 0xf2, 0xc7, 0x55, 0xf8, 0xf4,
00379    0xea, 0x4d, 0xb9, 0x72, 0xe5, 0x5e, 0xd9, 0x0f, 0xc9, 0xc8, 0xc8, 0xe0,
00380    0xd0, 0xa1, 0x43, 0x7c, 0x3d, 0x65, 0x02, 0x31, 0x91, 0xca, 0x29, 0xb0,
00381    0xa3, 0x6b, 0xd7, 0xa5, 0xbb, 0x30, 0xc4, 0xf9, 0x69, 0xa6, 0x8c, 0x76,
00382    0x05, 0xf2, 0x6d, 0x2d, 0xb8, 0xe1, 0x60, 0xc6, 0xca, 0xd8, 0x58, 0xee,
00383    0xc4, 0x29, 0x0f, 0xf9, 0x47, 0x8e, 0x1a, 0xc9, 0xd8, 0x31, 0x63, 0xa9,
00384    0x5d, 0xbb, 0xf6, 0x2b, 0xff, 0x5e, 0x19, 0x19, 0x19, 0x1c, 0x39, 0x7a,
00385    0x84, 0xf1, 0x5f, 0x8d, 0x27, 0x3a, 0x5a, 0xb9, 0xaf, 0xbf, 0xf8, 0xe2,
00386    0x0b, 0x46, 0x8e, 0x1c, 0x89, 0xa7, 0xa7, 0x5a, 0x1b, 0x5d, 0x11,
00387    0x80, 0x52, 0x95, 0x6f, 0xeb, 0xe8, 0xe8, 0xc8, 0x88, 0x11, 0x23, 0xf8,
00388    0xeb, 0xda, 0x5f, 0x1a, 0x15, 0xf8, 0xcb, 0xd1, 0x5f, 0xd0, 0xaa, 0x55,
00389    0x2b, 0x7e, 0xfb, 0xed, 0x37, 0x8d, 0x8b, 0x70, 0xcf, 0xed, 0xa4, 0xf9,
00390    0xf9, 0x9c, 0x3e, 0x7d, 0x9a, 0xae, 0x5d, 0xbb, 0xd2, 0xb7, 0x6f, 0x5f,
00391    0xc5, 0xe0, 0xef, 0x56, 0xc5, 0x83, 0x7d, 0x4d, 0x4d, 0xe0, 0xf3, 0xa7,
00392    0xd9, 0x32, 0xf8, 0xb5, 0xc0, 0x38, 0x4d, 0x45, 0xf3, 0x88, 0x14, 0x7e,
00393    0xb2, 0x29, 0xc7, 0x8c, 0x5a, 0x75, 0x15, 0x13, 0x83, 0xd6, 0xac, 0x5e,
00394    0x43, 0x9d, 0x3a, 0x75, 0xf0, 0x5d, 0xe1, 0x4b, 0x42, 0x42, 0xc2, 0xab,
00395    0x11, 0xa4, 0xfc, 0x7c, 0x4e, 0x9d, 0x3a, 0xc5, 0x07, 0xdd, 0x3e, 0xc0,
00396    0xa7, 0xb7, 0x8f, 0x62, 0xf0, 0x77, 0xee, 0xdc, 0x99, 0x73, 0xe7, 0xce,
00397    0xb1, 0x7c, 0xf9, 0x72, 0xc5, 0xe0, 0xd7, 0x25, 0x4a, 0x65, 0xc2, 0xbd,
00398    0x87, 0x87, 0x07, 0x8b, 0x17, 0x2f, 0xe6, 0xca, 0x95, 0x2b, 0x74, 0xe9,
00399    0xd2, 0x45, 0xad, 0x5d, 0x48, 0x48, 0x08, 0x1f, 0x7e, 0xf8, 0x21, 0xfd,
00400    0xfa, 0xf5, 0xe3, 0xfa, 0xf5, 0xeb, 0xda, 0x54, 0x63, 0x69, 0x24, 0x38,
00401    0x38, 0x98, 0x2f, 0xbe, 0xf8, 0x82, 0xf6, 0xed, 0xdb, 0x73, 0xf6, 0xec,
00402    0x59, 0xb5, 0x76, 0x0d, 0xdc, 0xdc, 0x58, 0xeb, 0xd5, 0x80, 0x99, 0x86,
00403    0x96, 0x54, 0x7d, 0xf0, 0x14, 0x84, 0x90, 0xd1, 0xfd, 0x1c, 0x58, 0x27,
00404    0x64, 0xf0, 0xd1, 0x53, 0x15, 0x47, 0x6a, 0xd7, 0x63, 0x58, 0xfd, 0x06,
```

```
00405 0x8a, 0xb6, 0xe3, 0xc6, 0x8e, 0xa3, 0x75, 0xdb, 0x36, 0xec, 0xdf, 0xbf,
00406 0x1f, 0x95, 0x4a, 0xf5, 0x52, 0xfe, 0xfe, 0xb3, 0x35, 0x9d, 0x21, 0x43,
00407 0x86, 0xf0, 0xce, 0x3b, 0xef, 0x70, 0xe6, 0xf4, 0x19, 0xb5, 0xb6, 0x35,
00408 0x6a, 0xd4, 0x60, 0xff, 0xfe, 0xfd, 0xec, 0xdb, 0xbf, 0x8f, 0xd6, 0xad,
00409 0x5b, 0x97, 0xaa, 0x6a, 0xc7, 0x52, 0x35, 0x05, 0x28, 0x8e, 0xec, 0xec,
00410 0x6c, 0x8e, 0x1c, 0x39, 0xc2, 0x94, 0x29, 0x53, 0xb8, 0x7f, 0xff, 0xbe,
00411 0xa2, 0xed, 0xd4, 0xa9, 0x53, 0x19, 0x36, 0x6c, 0x18, 0x1e, 0x1e, 0x1e,
00412 0xcf, 0xbf, 0x50, 0x15, 0x1f, 0xcf, 0xf6, 0x1d, 0x3b, 0x18, 0xff, 0xd5,
00413 0x57, 0x8a, 0x76, 0x66, 0x66, 0x66, 0x66, 0x4c, 0xae, 0x54, 0x85, 0x77, 0xd,
00414 0x8c, 0xb1, 0x4a, 0xc8, 0x92, 0x91, 0xfc, 0x32, 0x82, 0xd1, 0xd0, 0x80,
00415 0xe0, 0xca, 0x76, 0x6c, 0xce, 0xc8, 0xe0, 0xcf, 0x07, 0xca, 0x7d, 0xdc,
00416 0xa5, 0x6b, 0x17, 0x66, 0x4c, 0x9f, 0x41, 0xb3, 0x66, 0xcd, 0xfe, 0xf5,
00417 0x4a, 0xfb, 0xe3, 0xc7, 0x8f, 0xd9, 0xb8, 0x71, 0x23, 0x73, 0xe6, 0xcc,
00418 0x51, 0xb4, 0x33, 0x35, 0x35, 0x65, 0xd9, 0x0f, 0xcb, 0xf0, 0xe9, 0xed,
00419 0x83, 0x93, 0x93, 0xd3, 0x73, 0xfd, 0x0d, 0xb9, 0x06, 0xf0, 0x92, 0x04,
00420 0xe0, 0x19, 0x89, 0x89, 0x89, 0x6c, 0xdf, 0xb1, 0x9d, 0x71, 0x63, 0xc7,
00421 0x29, 0xda, 0x19, 0x19, 0x19, 0xb1, 0x6a, 0xd5, 0x2a, 0x7c, 0x7c, 0x7c,
00422 0xb0, 0xb7, 0xb7, 0xd7, 0xf8, 0xb9, 0x2a, 0x95, 0x8a, 0xc3, 0x47, 0x0e,
00423 0x33, 0x6e, 0xec, 0x38, 0x8d, 0x73, 0xbf, 0x61, 0xb5, 0xea, 0xd0, 0xd3,
00424 0xc2, 0x1a, 0x97, 0x88, 0x04, 0x19, 0xb5, 0xaf, 0x80, 0x3c, 0x47, 0x2b,
00425 0x2e, 0x99, 0x19, 0xb0, 0xe2, 0x49, 0x24, 0x0f, 0xd3, 0x94, 0xd3, 0x8a,
00426 0x27, 0x4c, 0x98, 0xc0, 0xe8, 0xd1, 0xa3, 0x9f, 0x4b, 0xec, 0x53, 0x52,
00427 0x52, 0xf0, 0xf3, 0xf3, 0x63, 0xee, 0xfc, 0xb9, 0x44, 0x45, 0x46, 0x29,
00428 0xda, 0x4e, 0x9c, 0x38, 0x91, 0x91, 0x23, 0x47, 0x52, 0xb5, 0x6a, 0xd5,
00429 0x7f, 0xf5, 0x5b, 0xa4, 0x00, 0xbc, 0x64, 0x01, 0x78, 0x46, 0x68, 0x68,
00430 0x28, 0xbe, 0xbe, 0xbe, 0xac, 0x5e, 0xad, 0xbc, 0x3d, 0xd3, 0xf4, 0x8d,
00431 0xa6, 0xcc, 0x9f, 0x3b, 0x9f, 0xb6, 0x6d, 0xdb, 0x62, 0x5a, 0x4c, 0x25,
00432 0x5a, 0x61, 0x61, 0x21, 0x57, 0xaf, 0x5e, 0x65, 0xde, 0xbc, 0x79, 0x1c,
00433 0x3d, 0x7a, 0x54, 0xf1, 0xb3, 0x3a, 0xd7, 0xa8, 0xc9, 0x60, 0x2b, 0x1b,
00434 0x6a, 0xc5, 0xa4, 0x61, 0x90, 0x2b, 0xb7, 0xf5, 0x5e, 0x35, 0xa9, 0x2e,
00435 0xd6, 0xfc, 0x6e, 0x0c, 0xab, 0xa3, 0x22, 0x49, 0x57, 0x12, 0x02, 0x43,
00436 0x58, 0xb7, 0x76, 0x1d, 0x1f, 0xf7, 0xfc, 0x58, 0xf1, 0x28, 0xb9, 0xec,
00437 0xec, 0x6c, 0x4e, 0x9d, 0x3e, 0xcd, 0xac, 0x39, 0xb3, 0xb8, 0x7e, 0x55,
00438 0xb9, 0x60, 0xe7, 0xe3, 0x8f, 0x3f, 0x66, 0xda, 0xb4, 0x69, 0x2f, 0xbc,
00439 0x97, 0x2f, 0x17, 0x01, 0x5f, 0x11, 0xb5, 0x6a, 0xd5, 0xc2, 0xd7, 0xd7,
00440 0x97, 0xb3, 0x67, 0xcf, 0xd2, 0xa1, 0xa3, 0xfa, 0xa3, 0x06, 0xaf, 0x5f,
00441 0xbb, 0x4e, 0xa7, 0x4e, 0x9d, 0x18, 0x38, 0x68, 0x20, 0x81, 0x81, 0x81,
00442 0x88, 0xff, 0x98, 0xa3, 0x3f, 0x7c, 0xf8, 0x90, 0x89, 0x93, 0x26, 0xd2,
00443 0xb2, 0x65, 0x4b, 0xc5, 0xe0, 0xaf, 0xe1, 0xe8, 0xc8, 0xf2, 0xda, 0x9e,
00444 0x7c, 0x9b, 0x63, 0x44, 0xed, 0x88, 0x24, 0x19, 0xfc, 0x25, 0x84, 0xdd,
00445 0xd3, 0x0c, 0x7c, 0xa2, 0x33, 0xf8, 0xc5, 0xa3, 0x3a, 0x7d, 0x6b, 0x29,
00446 0x1c, 0x3c, 0x5a, 0x08, 0xc3, 0x86, 0x0e, 0xa3, 0x6d, 0xfb, 0x76, 0x1c,
00447 0x39, 0x72, 0xe4, 0x1f, 0x8b, 0xc1, 0xcf, 0xce, 0xf7, 0xef, 0xdb, 0xb7,
00448 0x2f, 0x5d, 0xde, 0x7f, 0x5f, 0x31, 0xf8, 0x5b, 0xb5, 0x6a, 0xc5, 0xd1,
00449 0xdf, 0x8f, 0xb2, 0x7d, 0xfb, 0x76, 0x1a, 0x95, 0xa1, 0xd3, 0xa8, 0xcb,
00450 0xdc, 0x08, 0xe0, 0x3f, 0xc9, 0xcc, 0xcc, 0xe4, 0xe8, 0xd1, 0xa3, 0x4c,
00451 0x9c, 0x34, 0x91, 0x88, 0x47, 0x11, 0x8a, 0xb6, 0x33, 0x66, 0xce, 0xa0,
00452 0x8f, 0x4f, 0x1f, 0x4e, 0x9f, 0x39, 0xcd, 0xd4, 0x29, 0x53, 0x49, 0x53,
00453 0x7a, 0xb2, 0x18, 0x18, 0xf0, 0x8d, 0xb7, 0x17, 0x9d, 0xf2, 0x8c, 0xb1,
00454 0x89, 0x4d, 0x91, 0x11, 0xf9, 0x3a, 0xd7, 0x07, 0x6c, 0xcc, 0x09, 0x72,
00455 0xb0, 0xe0, 0xa7, 0xb8, 0x18, 0x2e, 0x6b, 0xd8, 0x36, 0xec, 0xed, 0xd3,
00456 0x9b, 0xa9, 0x53, 0xa6, 0x52, 0xbf, 0x7e, 0x7d, 0x1e, 0x3c, 0x78, 0xc0,
00457 0x4f, 0xeb, 0x7f, 0x62, 0xe9, 0xe2, 0xa5, 0x8a, 0xef, 0x71, 0x73, 0x73,
00458 0x63, 0xee, 0xdc, 0xb9, 0x7c, 0xfc, 0xf1, 0xc7, 0xd8, 0xda, 0xda, 0xbe,
00459 0xb4, 0xef, 0x2d, 0xa7, 0x00, 0x25, 0x20, 0x00, 0xcf, 0x88, 0x8d, 0x8b,
00460 0x65, 0xd3, 0xc6, 0xa2, 0x73, 0xd9, 0xc4, 0x0b, 0xae, 0xc6, 0x0f, 0xf2,
00461 0xf2, 0xe6, 0x13, 0x43, 0x53, 0x2a, 0x6a, 0x79, 0xcb, 0x8e, 0xa4, 0x64,
00462 0xc8, 0x71, 0xb0, 0xe4, 0x8c, 0x71, 0x21, 0xcb, 0xe2, 0x63, 0x79, 0x9a,
00463 0xa2, 0x2c, 0xca, 0xfd, 0xfa, 0xf5, 0x63, 0xc7, 0x8e, 0x1d, 0x1a, 0x3f,
00464 0x73, 0xde, 0xfc, 0x79, 0x0c, 0x1e, 0x34, 0x98, 0x8a, 0x15, 0x2b, 0xbe,
00465 0xf4, 0xef, 0x2b, 0xa7, 0x00, 0x25, 0x48, 0x85, 0xf2, 0x15, 0x98, 0x36,
00466 0x6d, 0x1a, 0x41, 0xb7, 0x83, 0x18, 0x30, 0x60, 0xc0, 0xbf, 0xfa, 0x8c,
00467 0x96, 0x2e, 0xe5, 0xf9, 0xb9, 0x5e, 0x03, 0xc6, 0x24, 0xe6, 0xc8, 0xe0,
00468 0xd7, 0x41, 0xcc, 0x92, 0xb3, 0xe8, 0x14, 0x9f, 0xcd, 0xae, 0x4a, 0x1e,
00469 0x7c, 0xe5, 0x55, 0x4f, 0xd1, 0x56, 0x53, 0xf0, 0x7f, 0xf6, 0xd9, 0x67,
00470 0x04, 0x05, 0x05, 0x31, 0x6d, 0xea, 0xb4, 0x57, 0x12, 0xfc, 0xba, 0x84,
00471 0xa1, 0x3e, 0x39, 0x89, 0x97, 0xa7, 0x17, 0x9b, 0x36, 0x6d, 0xe2, 0xe4,
00472 0xc9, 0x93, 0xb4, 0x69, 0xa3, 0x9d, 0xd0, 0xba, 0xb8, 0xb8, 0xb0, 0xa4,
00473 0x9e, 0x17, 0xcb, 0xcc, 0x1d, 0xa8, 0xff, 0x38, 0x15, 0xc3, 0xec, 0x3c,
00474 0x19, 0x6d, 0x3a, 0x8c, 0x43, 0x54, 0x0a, 0x03, 0x9f, 0xa4, 0xf3, 0x5b,
00475 0xd3, 0x37, 0xf8, 0xc8, 0xa3, 0xda, 0x73, 0xb7, 0x7d, 0xfb, 0xf6,
00476 0x9c, 0x39, 0x73, 0x86, 0xb5, 0x6b, 0xd7, 0xe2, 0xed, 0xed, 0x5d, 0x66,
00477 0x8f, 0xa2, 0xd3, 0x5b, 0x01, 0x00, 0x30, 0x36, 0x36, 0xa6, 0x7d, 0xfb,
00478 0xf6, 0x1c, 0x3e, 0x7c, 0x98, 0xcd, 0x5b, 0x36, 0x17, 0xbb, 0x03, 0xf0,
00479 0x9f, 0x3c, 0x7d, 0xfa, 0x94, 0xe8, 0x02, 0x03, 0x32, 0xed, 0xcc, 0x64,
00480 0x74, 0x95, 0x22, 0xdc, 0x1f, 0x3c, 0xc5, 0xc7, 0xc5, 0x85, 0xea, 0x5a,
00481 0xec, 0xcf, 0x57, 0xac, 0x58, 0x91, 0x5f, 0x7f, 0xfd, 0x95, 0x83, 0x07,
00482 0x0f, 0xf2, 0xf6, 0xdb, 0x6f, 0x97, 0x78, 0x7d, 0x81, 0x14, 0x80, 0xd7,
00483 0x80, 0x8d, 0x8d, 0x0d, 0x83, 0x07, 0x0d, 0x26, 0x2c, 0x2c, 0x4c, 0xa3,
00484 0xed, 0x0f, 0x77, 0x6f, 0x33, 0x38, 0x26, 0x9a, 0x13, 0x55, 0x1c, 0x51,
00485 0x59, 0x4b, 0x21, 0xd0, 0x75, 0x12, 0x3d, 0x9c, 0xd9, 0x50, 0xce, 0x94,
00486 0x5e, 0xd7, 0xae, 0xf0, 0x20, 0x31, 0x51, 0xb7, 0xf3, 0x1b, 0x1a, 0xb2,
00487 0x6c, 0xd9, 0x32, 0x6e, 0xfa, 0xdf, 0xe4, 0xe3, 0x8f, 0x3f, 0xc6, 0xca,
00488 0xca, 0x4a, 0xef, 0xda, 0xca, 0x58, 0xdf, 0x9d, 0x45, 0xdb, 0x39, 0xde,
00489 0x3c, 0x84, 0x78, 0x26, 0x26, 0xc4, 0xd3, 0xbc, 0x4a, 0x79, 0x46, 0xb8,
00490 0x55, 0xa0, 0x7e, 0x72, 0x16, 0x86, 0x2a, 0x39, 0x1d, 0xd0, 0x25, 0x32,
00491 0x1d, 0x2c, 0x39, 0x65, 0x61, 0x84, 0xef, 0xa3, 0x30, 0x12, 0x53, 0x34,
```

```
00492 0xef, 0xce, 0xec, 0xdb, 0xb7, 0x8f, 0xee, 0xdd, 0xbb, 0xeb, 0x75, 0x9b,
00493 0x19, 0x4b, 0xb7, 0x79, 0x3e, 0xae, 0x46, 0xc4, 0x71, 0x95, 0x38, 0xfa,
00494 0xd4, 0xaa, 0x49, 0x5f, 0x27, 0x07, 0x2a, 0x3d, 0x4e, 0xd6, 0xcd, 0x4b,
00495 0xd6, 0xf5, 0x88, 0x7c, 0x27, 0x2b, 0xae, 0x5b, 0x99, 0xb2, 0x32, 0xfe,
00496 0x09, 0x77, 0xc3, 0x9f, 0x6a, 0xfd, 0xbe, 0xd2, 0x7c, 0xc0, 0xac, 0x14,
00497 0x80, 0x57, 0xcc, 0x00, 0x8f, 0x6a, 0x1c, 0xcf, 0xc9, 0x22, 0x2e, 0xa6,
00498 0xf8, 0xbd, 0xe5, 0x5f, 0x42, 0xc3, 0xf8, 0x05, 0x98, 0xda, 0xa4, 0x09,
00499 0x9d, 0xd2, 0xf3, 0xb0, 0x7b, 0x9a, 0x26, 0x1b, 0xad, 0x84, 0x11, 0x36,
00500 0xe6, 0x84, 0x58, 0x99, 0xb2, 0x21, 0x2d, 0x91, 0x93, 0x0f, 0x9e, 0xc8,
00501 0x06, 0x91, 0x6b, 0x00, 0x2f, 0x8f, 0x5e, 0x89, 0xd9, 0x6c, 0x76, 0xaa,
00502 0xc0, 0xa7, 0x75, 0x95, 0xcb, 0x3a, 0x17, 0xde, 0xb8, 0xc1, 0xa7, 0x09,
00503 0x51, 0x9c, 0x2a, 0x6f, 0x41, 0xae, 0x8d, 0xb9, 0x6c, 0xb8, 0x12, 0xe2,
00504 0x49, 0x05, 0x3b, 0x56, 0xda, 0x41, 0x9f, 0xe0, 0x5b, 0x9c, 0x8c, 0x56,
00505 0x1f, 0xfc, 0xc6, 0x26, 0x26, 0xcc, 0xaa, 0xeb, 0xcd, 0x7b, 0xe5, 0x5d,
00506 0x65, 0xa3, 0xc9, 0x11, 0xc0, 0xf3, 0x60, 0x80, 0x6b, 0x54, 0x0a, 0x63,
00507 0xec, 0x2c, 0xe9, 0x50, 0xaf, 0x21, 0xeb, 0x13, 0x9e, 0x72, 0x46, 0xcd,
00508 0xbd, 0x77, 0xe1, 0x49, 0x49, 0x7c, 0x9d, 0x94, 0x44, 0x3b, 0xf7, 0x2a,
00509 0x7c, 0xee, 0x48, 0x48, 0xdd, 0x84, 0x4c, 0x0c, 0xb2, 0x72, 0x65, 0x13,
00510 0xbe, 0x02, 0xd2, 0x5c, 0xed, 0xf9, 0xc3, 0x58, 0xb0, 0x20, 0x28, 0x50,
00511 0xa3, 0xed, 0x48, 0x4f, 0x6f, 0x7a, 0x98, 0x9a, 0xe1, 0xf2, 0x28, 0x91,
00512 0xe0, 0xf4, 0x64, 0xd9, 0x78, 0x52, 0x00, 0xfe, 0x05, 0xa9, 0x59, 0x78,
00513 0xa6, 0x66, 0xb1, 0xc8, 0xce, 0x8e, 0x0b, 0x75, 0xed, 0x59, 0x95, 0x98,
00514 0x40, 0xf8, 0xd3, 0xe2, 0xe7, 0x99, 0xa7, 0x23, 0x23, 0x38, 0x1d, 0x19,
00515 0xc1, 0xe7, 0x9e, 0xde, 0x7c, 0x54, 0xde, 0x1a, 0xd7, 0x70, 0x99, 0x30,
00516 0xf4, 0xb2, 0xc8, 0x71, 0xb4, 0xe2, 0x82, 0x89, 0xe0, 0xc7, 0xb8, 0x48,
00517 0xc2, 0x93, 0x94, 0xdb, 0xb5, 0x6b, 0xd5, 0xea, 0x0c, 0xb4, 0xb0, 0xa4,
00518 0x66, 0xa2, 0x0a, 0x72, 0x32, 0x64, 0xe3, 0x49, 0x01, 0x78, 0x71, 0x4c,
00519 0x53, 0x55, 0xb4, 0x4f, 0x85, 0x26, 0x4e, 0x2e, 0x1c, 0xab, 0xe4, 0xc6,
00520 0xa2, 0x80, 0x5b, 0x08, 0x35, 0x07, 0x8c, 0x6c, 0xb8, 0x7b, 0x9b, 0xad,
00521 0xa6, 0xa6, 0x4c, 0xa8, 0x52, 0x8d, 0xce, 0x79, 0x06, 0xd8, 0xa4, 0xa8,
00522 0x64, 0x03, 0xfe, 0x4b, 0x0a, 0x4c, 0x8d, 0x08, 0x72, 0xb2, 0x60, 0x4b,
00523 0x46, 0x1a, 0x67, 0x1f, 0x2a, 0x1f, 0x21, 0xde, 0xd8, 0xd5, 0x95, 0xd1,
00524 0x15, 0x2a, 0xd2, 0x20, 0x29, 0x13, 0xa3, 0x68, 0x79, 0xe6, 0xa2, 0x14,
00525 0x80, 0x57, 0x80, 0x5d, 0x62, 0x06, 0xbd, 0x13, 0xe1, 0xad, 0x7a, 0x8d,
00526 0xd9, 0x99, 0x99, 0xc2, 0x2f, 0x6a, 0x0e, 0x21, 0xc9, 0xcd, 0xcd, 0x65,
00527 0x41, 0x58, 0x30, 0xfb, 0xca, 0x3b, 0x33, 0xc6, 0xbd, 0x12, 0xcd, 0xd2,
00528 0x72, 0x30, 0x4e, 0x91, 0x07, 0x84, 0x68, 0x8b, 0x10, 0x10, 0xe9, 0x51,
00529 0x8e, 0x1d, 0xe9, 0x49, 0xfc, 0x7a, 0xef, 0xb6, 0xa2, 0x6d, 0x95, 0xf2,
00530 0xe5, 0xf9, 0xc2, 0xdd, 0x9d, 0x96, 0x31, 0x69, 0x58, 0x86, 0xcb, 0xb3,
00531 0x18, 0xa4, 0x00, 0x94, 0x00, 0x95, 0x22, 0x12, 0x98, 0x60, 0x66, 0x4c,
00532 0xc7, 0x9a, 0x9e, 0xac, 0xcb, 0x4a, 0xe3, 0xca, 0x93, 0xe2, 0x0f, 0x90,
00533 0x08, 0x8e, 0x8b, 0x67, 0x54, 0x5c, 0x3c, 0x9d, 0xab, 0x57, 0xe5, 0xb3,
00534 0x2a, 0xe5, 0xa8, 0x19, 0x9d, 0x0c, 0x79, 0x05, 0xb2, 0x01, 0x15, 0x48,
00535 0x70, 0xb5, 0xe3, 0xb0, 0x69, 0x1e, 0xbe, 0x81, 0xd7, 0x15, 0xed, 0x8c,
00536 0x4c, 0x8d, 0x18, 0xe3, 0x51, 0x93, 0xae, 0x06, 0xa6, 0x38, 0x86, 0xc5,
00537 0xcb, 0x86, 0x93, 0x02, 0x50, 0xb2, 0x18, 0xe6, 0xe4, 0xd3, 0x28, 0x3e,
00538 0x9f, 0x1f, 0x5c, 0x6d, 0xb8, 0xe8, 0xdc, 0x84, 0x15, 0x8f, 0xc3, 0x89,
00539 0x4c, 0x2c, 0x7e, 0x7e, 0x7a, 0xec, 0x41, 0x38, 0xc7, 0x08, 0x67, 0x58,
00540 0xed, 0xda, 0x7c, 0x54, 0x68, 0x41, 0xf9, 0x78, 0x39, 0x37, 0xfd, 0x5f,
00541 0xb2, 0xec, 0x2c, 0x38, 0x6b, 0x69, 0xc4, 0xaa, 0x84, 0x68, 0xa2, 0xe3,
00542 0x95, 0x03, 0xba, 0x6f, 0xdd, 0xda, 0xf8, 0x98, 0x5a, 0x53, 0x39, 0x22,
00543 0x11, 0xc8, 0x96, 0x8d, 0x27, 0x05, 0xe0, 0xf5, 0x61, 0x1e, 0xa3, 0xe2,
00544 0x1d, 0x54, 0x34, 0xae, 0x52, 0x95, 0x03, 0xe5, 0xca, 0xb3, 0x32, 0x2c,
00545 0x44, 0xed, 0x01, 0xa4, 0xeb, 0x42, 0x42, 0xd8, 0x61, 0x61, 0xc1, 0x14,
00546 0x77, 0x0f, 0xde, 0x31, 0xb2, 0xc0, 0x22, 0x5a, 0x2e, 0x14, 0xe6, 0x5b,
00547 0x9b, 0xe2, 0x5f, 0xce, 0x8a, 0xd5, 0x4f, 0xe3, 0x08, 0xb8, 0xa7, 0x7c,
00548 0xec, 0x5a, 0xbb, 0x2a, 0x1e, 0x7c, 0x6a, 0x69, 0x8b, 0x57, 0x62, 0x26,
00549 0x86, 0xb9, 0x89, 0xd2, 0xf9, 0xa4, 0x00, 0xe8, 0x0e, 0x0e, 0x11, 0x89,
00550 0x0c, 0x36, 0x34, 0xa0, 0x4d, 0x5d, 0x6f, 0xb6, 0xa4, 0x26, 0x73, 0x28,
00551 0xaa, 0xf8, 0x4b, 0x4e, 0x33, 0x55, 0x2a, 0x66, 0x84, 0xdc, 0x63, 0x7f,
00552 0xc5, 0x4a, 0x8c, 0xa8, 0xe2, 0x4c, 0xa3, 0xe8, 0x34, 0x8c, 0xf5, 0x70,
00553 0x5a, 0x20, 0x8c, 0x0d, 0x09, 0x2d, 0x67, 0xc5, 0x8e, 0x7c, 0x15, 0x87,
00554 0x02, 0xfc, 0x15, 0x6d, 0x6b, 0x56, 0x75, 0x61, 0x94, 0x65, 0x79, 0x5a,
00555 0x66, 0x16, 0x60, 0x1a, 0x23, 0x93, 0xae, 0xa4, 0x00, 0xe8, 0x2a, 0x85,
00556 0x82, 0x6a, 0x4f, 0xd2, 0x98, 0x65, 0x67, 0xc3, 0x07, 0xf5, 0x1a, 0xb0,
00557 0x31, 0x2d, 0x85, 0xab, 0x11, 0xc5, 0x9f, 0x46, 0x74, 0xf3, 0x49, 0x14,
00558 0xc3, 0x9e, 0x44, 0xf1, 0x61, 0xed, 0x3a, 0xf4, 0x33, 0xb0, 0xa0, 0x7a,
00559 0x7c, 0x16, 0x14, 0x14, 0xea, 0x45, 0x33, 0xc5, 0x56, 0xb2, 0xe7, 0xa0,
00560 0x11, 0xac, 0x09, 0x54, 0xbe, 0xe9, 0xc9, 0xdc, 0xdc, 0x9c, 0x89, 0x35,
00561 0xeb, 0xd0, 0x21, 0x2b, 0x1f, 0xdb, 0x27, 0x72, 0x65, 0x5f, 0x0a, 0x40,
00562 0x29, 0xc1, 0x28, 0x35, 0x8b, 0x66, 0xa9, 0xe0, 0xe5, 0x64, 0xc3, 0x19,
00563 0xaf, 0x7a, 0x2c, 0x8a, 0x8a, 0x24, 0x2d, 0xb5, 0xf8, 0x0b, 0x40, 0xf7,
00564 0x87, 0x04, 0xb3, 0x1f, 0x18, 0x5b, 0xab, 0x0e, 0xdd, 0xcc, 0xcd, 0x71,
00565 0x8c, 0x2c, 0xbb, 0xc7, 0x8c, 0x65, 0x54, 0x72, 0xe2, 0x58, 0x81, 0x0a,
00566 0xdf, 0xf0, 0x30, 0x32, 0x34, 0x5c, 0xe8, 0x3a, 0xac, 0x4e, 0x1d, 0xba,
00567 0xe7, 0x19, 0xe2, 0xf6, 0x58, 0x1e, 0xbb, 0x26, 0x05, 0xa0, 0x94, 0x62,
00568 0x99, 0x98, 0xc1, 0xfb, 0x40, 0xd3, 0x0a, 0x55, 0xd8, 0xe7, 0x9c, 0xc5,
00569 0x5a, 0x85, 0xbb, 0x0b, 0x7c, 0x43, 0x83, 0xd9, 0xeb, 0xec, 0xcc, 0xd8,
00570 0x6a, 0x55, 0x78, 0x2b, 0x21, 0x03, 0x8b, 0xb4, 0xb2, 0xb3, 0xb8, 0x95,
00571 0x6b, 0x6f, 0xce, 0x65, 0x0b, 0x13, 0x56, 0x27, 0x46, 0x13, 0x1a, 0x13,
00572 0xa3, 0x68, 0xfb, 0xae, 0x5b, 0x45, 0x3e, 0x77, 0x2e, 0x4f, 0x8d, 0xe8,
00573 0x54, 0x0c, 0xe4, 0x8e, 0x89, 0x14, 0x80, 0xb2, 0x80, 0x4b, 0x5c, 0x1a,
00574 0xc3, 0xd0, 0xd0, 0x68, 0xef, 0xdd, 0x88, 0x9d, 0x05, 0xd9, 0x1c, 0xb8,
00575 0x77, 0xaf, 0x58, 0xbb, 0xa8, 0xf8, 0x78, 0x26, 0xc6, 0xc7, 0xd3, 0xbc,
00576 0x9a, 0x2b, 0x23, 0x2a, 0xba, 0x50, 0x3f, 0x59, 0x85, 0x61, 0x29, 0x4e,
00577 0x2b, 0x16, 0xf6, 0x96, 0xdc, 0xb1, 0x33, 0x67, 0x73, 0x6a, 0x12, 0xa7,
00578 0xee, 0x29, 0x5f, 0xfa, 0xd9, 0xa8, 0xa2, 0x1b, 0x23, 0x9c, 0xca, 0xd3,
```

```
00579 0x28, 0x21, 0x13, 0x93, 0x08, 0xb9, 0x38, 0x2a, 0x05, 0xa0, 0x8c, 0x61,
00580 0x50, 0x28, 0xa8, 0x15, 0x95, 0xcc, 0x74, 0x27, 0x73, 0x3a, 0x7b, 0x35,
00581 0x60, 0x65, 0x42, 0x2c, 0x77, 0xe3, 0xe2, 0x8a, 0xb5, 0xbd, 0xfa, 0x30,
00582 0x86, 0xab, 0xc4, 0xd0, 0xcb, 0xa3, 0x2a, 0xfd, 0x1d, 0x1d, 0xa8, 0x5c,
00583 0x0a, 0xcb, 0x8e, 0x1f, 0xbb, 0x97, 0x63, 0x67, 0x6a, 0x22, 0xbb, 0x02,
00584 0xef, 0x2a, 0xcf, 0xf3, 0x4d, 0x4d, 0x99, 0x51, 0xbb, 0x36, 0x6d, 0xb3,
00585 0x04, 0x96, 0x91, 0x32, 0x67, 0x5f, 0x0a, 0x40, 0x59, 0x6f, 0xf4, 0xc4,
00586 0x6c, 0xde, 0x24, 0x1b, 0x6f, 0xe7, 0x72, 0x1c, 0xb3, 0xb1, 0x67, 0x43,
00587 0x7a, 0x0a, 0x71, 0x6a, 0x84, 0xc0, 0xef, 0x51, 0x38, 0x7e, 0xc0, 0x94,
00588 0xa6, 0x4d, 0xe9, 0x9c, 0x91, 0x87, 0x5d, 0x6c, 0xaa, 0xce, 0xff, 0xbe,
00589 0x44, 0x67, 0x6b, 0x8e, 0xd9, 0x1a, 0xf3, 0xfd, 0x8d, 0xeb, 0x1a, 0x6d,
00590 0xbf, 0xac, 0x5e, 0x8b, 0x1e, 0x96, 0xb6, 0x38, 0x3e, 0x96, 0x19, 0x7c,
00591 0x52, 0x00, 0xf4, 0x0c, 0xeb, 0xf8, 0x4c, 0x3e, 0x06, 0xde, 0x72, 0x2b,
00592 0xcf, 0xaf, 0x76, 0xf6, 0x6c, 0x0e, 0x0d, 0x51, 0x6b, 0xfb, 0xdd, 0xf5,
00593 0xeb, 0xf8, 0x39, 0x3a, 0x31, 0xca, 0xd5, 0x85, 0xd6, 0x99, 0x60, 0x9a,
00594 0xa6, 0x7b, 0xf5, 0x05, 0x2a, 0x07, 0x4b, 0x2e, 0x3a, 0x58, 0xb2, 0xf8,
00595 0x41, 0x18, 0xf1, 0x61, 0xca, 0x42, 0xd5, 0xbb, 0x6e, 0x6d, 0x7c, 0x4c,
00596 0x2c, 0xf1, 0x78, 0x9c, 0x02, 0x89, 0x32, 0xf8, 0xa5, 0x00, 0xe8, 0x31,
00597 0xae, 0xd1, 0x69, 0x8c, 0xb1, 0xb7, 0xa4, 0x63, 0xfd, 0x86, 0x6c, 0x48,
00598 0x4a, 0xe4, 0x94, 0x9a, 0xfc, 0x81, 0x87, 0x49, 0x89, 0x4c, 0x48, 0x4a,
00599 0xa4, 0x4d, 0x05, 0x57, 0x86, 0x55, 0x2a, 0x8f, 0x67, 0xb2, 0x0a, 0x83,
00600 0xcc, 0x9c, 0xd7, 0xfe, 0xfd, 0x0b, 0xec, 0x2c, 0xb8, 0x69, 0x69, 0xc8,
00601 0x9a, 0xa4, 0x04, 0xfc, 0xc3, 0x95, 0x87, 0xfb, 0x6f, 0xb9, 0x57, 0xe1,
00602 0x33, 0x33, 0x2b, 0x1a, 0xa6, 0xe5, 0x63, 0x98, 0x29, 0x87, 0xfb, 0x52,
00603 0x00, 0x24, 0x45, 0xa4, 0x64, 0x51, 0x37, 0x25, 0x8b, 0x85, 0xf6, 0x36,
00604 0x5c, 0xf4, 0xf4, 0x66, 0x55, 0x7c, 0x1c, 0x0f, 0xd5, 0xa4, 0xc3, 0x9e,
00605 0x8b, 0x8d, 0xe1, 0x5c, 0x6c, 0x0c, 0x43, 0xbc, 0xeb, 0xd1, 0xb3, 0xbc,
00606 0x35, 0xae, 0x0f, 0x5f, 0x4f, 0x46, 0x9c, 0x30, 0x31, 0xe2, 0xbe, 0x93,
00607 0x25, 0x3b, 0xf2, 0xb3, 0x38, 0x70, 0xef, 0x81, 0xa2, 0x6d, 0x45, 0x6b,
00608 0x1b, 0x26, 0xd4, 0xaa, 0x45, 0xcb, 0xf4, 0x3c, 0x4c, 0xe3, 0x65, 0x22,
00609 0x8f, 0x14, 0x00, 0x49, 0xb1, 0x98, 0xa6, 0x64, 0xd1, 0x2e, 0x05, 0x1a,
00610 0xdb, 0x97, 0xe3, 0x98, 0xab, 0x2b, 0xdf, 0xdf, 0xbb, 0x47, 0x7e, 0x5e,
00611 0xf1, 0x87, 0x8f, 0x6e, 0xbc, 0x1d, 0xc4, 0x0e, 0x73, 0x0b, 0xc6, 0x57,
00612 0xa9, 0x4a, 0xa7, 0x9c, 0x42, 0x6c, 0x53, 0x4a, 0x6e, 0xdb, 0xf0, 0x69,
00613 0x15, 0x27, 0xf6, 0xab, 0x32, 0xf9, 0x29, 0x38, 0x48, 0xd1, 0xce, 0xc4,
00614 0xd4, 0x94, 0x49, 0x9e, 0xde, 0x74, 0xcc, 0xce, 0xc3, 0xee, 0xa1, 0x4c,
00615 0xdd, 0x95, 0x02, 0x20, 0xd1, 0x0a, 0xbb, 0x14, 0x15, 0xbd, 0x53, 0x54,
00616 0xb4, 0xf2, 0x6c, 0xc0, 0xae, 0xfc, 0x4c, 0xb6, 0xdf, 0x29, 0x7e, 0xdb,
00617 0x30, 0x3b, 0x5b, 0xc5, 0x82, 0x90, 0xbb, 0xec, 0x29, 0xe7, 0xc4, 0xd8,
00618 0xca, 0x95, 0x79, 0x23, 0x3d, 0xf7, 0x95, 0x96, 0x1d, 0x67, 0xb8, 0xd8,
00619 0x72, 0xc2, 0x28, 0x9f, 0x6f, 0x83, 0xfc, 0xd5, 0x9e, 0x87, 0xf0, 0x8c,
00620 0x41, 0xb5, 0x6a, 0xd1, 0xcb, 0xc8, 0x02, 0xb7, 0x47, 0x72, 0x4b, 0x4f,
00621 0x0a, 0x80, 0xe4, 0x5f, 0x51, 0x31, 0x22, 0x81, 0xaf, 0x2c, 0x4c, 0x78,
00622 0xa7, 0x6e, 0x7d, 0xd6, 0x26, 0xc7, 0x73, 0x25, 0xb6, 0xf8, 0x24, 0x9a,
00623 0xd0, 0x84, 0x44, 0x46, 0x27, 0x24, 0xd2, 0xc9, 0xc3, 0x83, 0x21, 0x1e,
00624 0xe5, 0xa8, 0x19, 0x9b, 0x0a, 0x2f, 0xf1, 0x16, 0xa3, 0x5c, 0x4b, 0x53,
00625 0xae, 0xda, 0x98, 0xb0, 0x21, 0x23, 0x91, 0x5b, 0x4f, 0x94, 0x0f, 0xe0,
00626 0xec, 0x54, 0xb5, 0x1a, 0x83, 0x6d, 0xec, 0xa8, 0x93, 0x92, 0x0d, 0x69,
00627 0xa9, 0xb2, 0x13, 0xa5, 0x00, 0x48, 0x5e, 0x04, 0x43, 0x55, 0x1e, 0x0d,
00628 0x55, 0x79, 0xfc, 0x50, 0xc1, 0x8e, 0x0b, 0xe5, 0x5c, 0xf8, 0xf1, 0x49,
00629 0x24, 0x8f, 0x92, 0x8b, 0x5f, 0x40, 0xfb, 0xe3, 0xd1, 0x23, 0xfe, 0xe0,
00630 0x11, 0x43, 0x6b, 0xd7, 0xa6, 0xa7, 0x8d, 0xf5, 0x0b, 0x97, 0x1d, 0x17,
00631 0x9a, 0x18, 0x12, 0x52, 0xd1, 0x81, 0x0d, 0xa9, 0x4f, 0x39, 0x15, 0x12,
00632 0xaa, 0x68, 0xdb, 0xc8, 0xbd, 0x0a, 0x43, 0xed, 0x1d, 0x69, 0xfa, 0x34,
00633 0x1d, 0x93, 0x64, 0xb9, 0xc0, 0x27, 0x05, 0x40, 0xf2, 0x52, 0x31, 0x8f,
00634 0xcd, 0xa6, 0x03, 0xd9, 0x34, 0x71, 0xaf, 0xca, 0x61, 0xb7, 0x4a, 0xfc,
00635 0x70, 0xf7, 0xb6, 0xda, 0xdb, 0x8e, 0xd7, 0x87, 0x84, 0xe0, 0x67, 0x6d,
00636 0xcd, 0x57, 0x15, 0xdc, 0xe8, 0x68, 0x6c, 0x89, 0xe5, 0xf3, 0x5e, 0x63,
00637 0x2e, 0xe0, 0xb1, 0x8b, 0x35, 0x7b, 0xc9, 0xe5, 0xe7, 0x80, 0x1b, 0x8a,
00638 0xa6, 0x15, 0x9c, 0x9d, 0x19, 0x51, 0xc1, 0x89, 0x0e, 0xb9, 0xa6, 0x58,
00639 0xc9, 0x44, 0x1e, 0x29, 0x00, 0x92, 0x57, 0x8b, 0x43, 0x64, 0x12, 0x03,
00640 0x0c, 0xa0, 0x95, 0x67, 0x7d, 0xb6, 0xa4, 0x24, 0x71, 0xf0, 0x49, 0xf1,
00641 0xdb, 0x86, 0xa9, 0x19, 0x19, 0xcc, 0xbe, 0x1f, 0xca, 0x81, 0x4a, 0x95,
00642 0x18, 0x59, 0xd5, 0x85, 0xc6, 0xd1, 0x69, 0x18, 0xe5, 0xe4, 0x6b, 0xfc,
00643 0xfc, 0xe4, 0x72, 0xd6, 0xfc, 0x69, 0x6f, 0xc6, 0x77, 0x37, 0x6e, 0x6a,
00644 0xbc, 0x4e, 0x7d, 0x8c, 0x97, 0x37, 0x5d, 0x0b, 0x8c, 0x70, 0x7e, 0x9c,
00645 0x8a, 0x3c, 0x98, 0x43, 0x0a, 0x80, 0xa4, 0xa4, 0x10, 0x50, 0xf5, 0x49,
00646 0x2a, 0x33, 0xed, 0xac, 0xe9, 0x5a, 0xc7, 0x9b, 0x0d, 0xe9, 0x29, 0x5c,
00647 0x53, 0x73, 0x2c, 0x99, 0x7f, 0x54, 0x14, 0xc3, 0xa2, 0xa2, 0xf8, 0xa8,
00648 0xae, 0x27, 0x7d, 0x0c, 0xcc, 0xa9, 0x1e, 0x97, 0x85, 0x41, 0x31, 0x65,
00649 0xc7, 0xd9, 0xb6, 0xe6, 0x5c, 0xb4, 0x37, 0x67, 0x79, 0xec, 0x63, 0xa2,
00650 0xee, 0x2b, 0xaf, 0xd8, 0x7f, 0xe2, 0x59, 0x8b, 0xde, 0x06, 0xe6, 0x54,
00651 0x8f, 0x49, 0x87, 0x42, 0x21, 0xfb, 0x43, 0x0a, 0x80, 0xe4, 0x75, 0x60,
00652 0x94, 0xaa, 0x2a, 0x2a, 0x3b, 0x76, 0xb4, 0xe3, 0x03, 0x67, 0x96,
00653 0x3d, 0x79, 0x4c, 0x7c, 0x42, 0xf1, 0x99, 0x75, 0xfb, 0xee, 0xdd, 0x65,
00654 0x1f, 0xf0, 0x45, 0xf5, 0x9a, 0xf4, 0xb0, 0xb4, 0xc1, 0xe9, 0x71, 0xd1,
00655 0x02, 0x5d, 0x81, 0xb9, 0x31, 0xb7, 0x5c, 0xac, 0x59, 0x97, 0x9c, 0xc8,
00656 0x95, 0x5b, 0xca, 0x05, 0x3b, 0x6f, 0xba, 0x57, 0x66, 0xa8, 0x8d, 0x23,
00657 0x0d, 0x12, 0xb3, 0x30, 0xca, 0x96, 0xfb, 0xf9, 0x52, 0x00, 0x24, 0x3a,
00658 0x81, 0x65, 0x52, 0x26, 0x9d, 0x93, 0x32, 0x69, 0xe2, 0x5a, 0x89, 0xfd,
00659 0x8e, 0x8e, 0xac, 0x09, 0x55, 0xbf, 0x60, 0xb7, 0xea, 0x41, 0x18, 0x7b,
00660 0x9d, 0xca, 0x31, 0xbe, 0xaa, 0x07, 0x55, 0xf2, 0x0b, 0xd9, 0x93, 0x9f,
00661 0x85, 0xdf, 0x2d, 0xe5, 0x83, 0x39, 0x2a, 0xd9, 0x31, 0xb6, 0x72,
00662 0x15, 0x5a, 0xa5, 0xe7, 0x61, 0xfe, 0x44, 0xae, 0xec, 0x4b, 0x01, 0x90,
00663 0xe8, 0x24, 0xce, 0x31, 0x29, 0x0c, 0x35, 0x36, 0xa4, 0x6d, 0x9d, 0x7a,
00664 0xec, 0x34, 0xc8, 0xe3, 0x80, 0x9a, 0x12, 0xdc, 0x98, 0xc4, 0x04, 0x26,
00665 0x6a, 0x91, 0x83, 0x6f, 0x60, 0x68, 0xc0, 0xc4, 0xda, 0x9e, 0x74, 0x2e,
```

```
00666 0x30, 0xc0, 0x21, 0x4a, 0x3e, 0xf1, 0xa5, 0x00, 0x48, 0x74, 0x1e, 0x83,
00667 0xfc, 0x42, 0x6a, 0xc5, 0xa6, 0xf3, 0x8d, 0xad, 0x29, 0xef, 0x79, 0x37,
00668 0x64, 0x75, 0x4a, 0x02, 0xb7, 0xa2, 0xa2, 0x9e, 0xfb, 0x73, 0xfa, 0x55,
00669 0xaf, 0x46, 0x1f, 0x0b, 0x7b, 0x2a, 0x46, 0xc9, 0x44, 0x1e, 0x29, 0x00,
00670 0x92, 0x52, 0x87, 0x49, 0x5a, 0x2e, 0xcd, 0xd3, 0x72, 0xf1, 0xae, 0xe0,
00671 0xc8, 0x1f, 0x35, 0xac, 0x58, 0x9d, 0xf8, 0x94, 0x44, 0x2d, 0xf6, 0xe7,
00672 0x3b, 0x56, 0xad, 0xc6, 0x20, 0x1b, 0x5b, 0xea, 0x46, 0xa7, 0x61, 0x98,
00673 0x28, 0x83, 0x5f, 0x0a, 0x80, 0xa4, 0x54, 0x63, 0x15, 0x9b, 0xc6, 0x47,
00674 0x40, 0x8b, 0x0a, 0x95, 0xf0, 0x73, 0x74, 0x62, 0xcb, 0x83, 0xe2, 0x8f,
00675 0x25, 0xab, 0x5b, 0xa1, 0x02, 0x23, 0x6c, 0x1d, 0x69, 0x91, 0x23, 0x30,
00676 0x89, 0x94, 0xe7, 0xf0, 0xe9, 0x03, 0xf2, 0x7a, 0x70, 0x3d, 0xc2, 0x35,
00677 0x36, 0x9d, 0xb1, 0x79, 0x26, 0x2c, 0xa9, 0x55, 0xeb, 0x1f, 0xff, 0xd7,
00678 0xce, 0xc6, 0x96, 0xf5, 0xce, 0x6e, 0xb4, 0x79, 0x9a, 0x85, 0x49, 0xaa,
00679 0xbc, 0xcb, 0x50, 0x8e, 0x00, 0x24, 0x65, 0x93, 0x34, 0x15, 0xe5, 0x12,
00680 0xfe, 0x59, 0x2c, 0xe4, 0x98, 0x99, 0x83, 0xd5, 0x63, 0x39, 0xdc, 0x97,
00681 0x23, 0x00, 0x89, 0x44, 0x22, 0x05, 0x40, 0x22, 0x91, 0x48, 0x01, 0x90,
00682 0x48, 0x24, 0x52, 0x00, 0x24, 0x12, 0x89, 0x14, 0x00, 0x89, 0x44, 0x22,
00683 0x05, 0x40, 0x22, 0x91, 0x48, 0x01, 0x90, 0x48, 0x24, 0x52, 0x00, 0x24,
00684 0x12, 0x89, 0x14, 0x00, 0x89, 0x44, 0x22, 0x05, 0x40, 0x22, 0x91, 0x48,
00685 0x01, 0x90, 0x48, 0x24, 0x52, 0x00, 0x24, 0x12, 0x89, 0x14, 0x00, 0x89,
00686 0x44, 0x22, 0x05, 0x40, 0x22, 0x91, 0x48, 0x01, 0x90, 0x48, 0x24, 0x52,
00687 0x00, 0x24, 0x12, 0x89, 0x14, 0x00, 0x89, 0x44, 0x22, 0x05, 0x40, 0x22,
00688 0x91, 0x48, 0x01, 0x90, 0x48, 0x24, 0x52, 0x00, 0x24, 0x12, 0x89, 0x14,
00689 0x00, 0x89, 0x44, 0x22, 0x05, 0x40, 0x22, 0x91, 0x48, 0x01, 0x90, 0x48,
00690 0x24, 0x52, 0x00, 0x24, 0x12, 0x89, 0x14, 0x00, 0x89, 0x44, 0x22, 0x05,
00691 0x40, 0x22, 0x91, 0x48, 0x01, 0x90, 0x48, 0x24, 0x52, 0x00, 0x24, 0x12,
00692 0x89, 0x44, 0x00, 0x89, 0x44, 0x22, 0x05, 0x40, 0x1b, 0xd2, 0xd3, 0xd3,
00693 0xa5, 0x17, 0x48, 0xa4, 0x00, 0xe8, 0x1b, 0x79, 0x79, 0x79, 0x9c, 0x3c,
00694 0x79, 0x92, 0x1e, 0x3d, 0x7a, 0x14, 0xfb, 0xff, 0x77, 0x2b, 0xda, 0x91,
00695 0x67, 0x63, 0x2e, 0x3d, 0xa4, 0x0c, 0x10, 0xe9, 0x6c, 0x4d, 0x54, 0x79,
00696 0x3b, 0xd9, 0x10, 0x52, 0x00, 0x40, 0x08, 0x41, 0x50, 0x50, 0x10, 0x83,
00697 0x07, 0x0f, 0xa6, 0x43, 0x87, 0x0e, 0x9c, 0x3d, 0x7b, 0xb6, 0x58, 0xbb,
00698 0xc9, 0xc1, 0x77, 0x99, 0x6d, 0x63, 0x40, 0x88, 0x47, 0x39, 0x0a, 0x0d,
00699 0x0d, 0xa4, 0xa7, 0x94, 0x42, 0x52, 0xcb, 0x59, 0xb3, 0xb3, 0x82, 0x19,
00700 0x7d, 0x63, 0x1e, 0x71, 0x31, 0x3a, 0xee, 0x1f, 0xff, 0xff, 0xf3, 0xd6,
00701 0x9f, 0xb9, 0x7d, 0xfb, 0x36, 0x42, 0x08, 0x29, 0x00, 0xfa, 0xc0, 0xe3,
00702 0xa8, 0xc7, 0xcc, 0x9d, 0x37, 0x97, 0xfa, 0xf5, 0xeb, 0xb3, 0x73, 0xe7,
00703 0x4e, 0x8d, 0xf6, 0x47, 0xef, 0xde, 0xc3, 0x27, 0xe0, 0x3a, 0xab, 0x9d,
00704 0x8d, 0x88, 0x2b, 0x6f, 0x2b, 0x23, 0xaa, 0xb4, 0x8c, 0xee, 0x9c, 0x6d,
00705 0x39, 0x5f, 0xd5, 0x91, 0x4f, 0xe3, 0x22, 0x58, 0x12, 0x1c, 0x42, 0x66,
00706 0x46, 0x46, 0xb1, 0x76, 0xdb, 0xb6, 0x6e, 0xa3, 0x5e, 0xbd, 0x7a, 0x2c,
00707 0x5b, 0xb6, 0x8c, 0xc4, 0xc4, 0x44, 0x29, 0x00, 0x65, 0x95, 0xb4, 0xb4,
00708 0x34, 0x36, 0x6e, 0xdc, 0x48, 0xe3, 0x46, 0x8d, 0x99, 0x35, 0x73, 0xd6,
00709 0x73, 0xbf, 0x7f, 0x63, 0xc8, 0x7d, 0xba, 0x86, 0x07, 0x73, 0xa0, 0xa6,
00710 0x03, 0x75, 0xbe, 0x00, 0x00, 0x15, 0x4e, 0x49, 0x44, 0x41, 0x54, 0x82,
00711 0x25, 0x99, 0xe5, 0xac, 0x65, 0x84, 0xe9, 0x28, 0x85, 0x86, 0x06, 0xdc,
00712 0xab, 0x64, 0xc7, 0x94, 0xdc, 0x34, 0xc6, 0xf8, 0xdf, 0x24, 0x5c, 0xcb,
00713 0xf5, 0x9d, 0x09, 0x13, 0x26, 0xd0, 0xe6, 0xed, 0x36, 0x1c, 0x3a, 0x72,
00714 0x98, 0xdc, 0xdc, 0x5c, 0x29, 0x00, 0x65, 0x85, 0xdc, 0xdc, 0x5c, 0x8e,
00715 0x1f, 0x3f, 0x4e, 0xc7, 0x8e, 0x1d, 0xf9, 0xfc, 0xf3, 0xcf, 0x49, 0x48,
00716 0x48, 0x50, 0x6b, 0xdb, 0xbf, 0x7f, 0x86, 0x0e, 0x1d, 0xaa, 0xf6,
00717 0xff, 0xf3, 0x73, 0x73, 0x99, 0x1d, 0x7c, 0x97, 0x31, 0xb9, 0x69, 0x5c,
00718 0xf3, 0x70, 0xa4, 0xc0, 0xce, 0x42, 0x46, 0x9c, 0x0e, 0x11, 0xeb, 0x6a,
00719 0xc7, 0x4a, 0x07, 0x63, 0xfa, 0xde, 0x0e, 0x4, 0x54, 0xc4, 0xa3, 0xe7,
00720 0x7e, 0xff, 0xdd, 0x3b, 0x77, 0xe9, 0xd6, 0xf5, 0x03, 0x06, 0x0d, 0x1a,
00721 0xc4, 0x9d, 0x3b, 0x77, 0xf4, 0x66, 0x5a, 0x50, 0x26, 0x05, 0x40, 0x08,
00722 0x41, 0x60, 0x60, 0x20, 0x03, 0x07, 0x0e, 0xa4, 0x53, 0xa7, 0x4e, 0x5c,
00723 0xbb, 0x76, 0x4d, 0xad, 0x6d, 0xd3, 0xa6, 0x4d, 0x39, 0x7e, 0xfc, 0x38,
00724 0x9b, 0x36, 0x6d, 0x62, 0xf5, 0xea, 0xd5, 0x9c, 0x39, 0x7b, 0x86, 0xb6,
00725 0x6d, 0xdb, 0xaa, 0xb5, 0xbf, 0x19, 0xc9, 0xf0, 0x80, 0x9b, 0xcc,
00726 0x32, 0xcc, 0xe0, 0x41, 0x65, 0x47, 0x84, 0x5c, 0x1f, 0x78, 0xad, 0x64,
00727 0xb9, 0x39, 0xb0, 0xdf, 0xcd, 0x8a, 0x0f, 0xee, 0xdf, 0x65, 0xcb, 0x83,
00728 0x30, 0x45, 0xdb, 0x65, 0xcb, 0x96, 0xf1, 0xfb, 0xef, 0xbf, 0xd3, 0xa4,
00729 0x49, 0x13, 0xb5, 0x36, 0xbb, 0x76, 0xed, 0xc2, 0xdb, 0xdb, 0x9b, 0xc5,
00730 0xdf, 0x2f, 0xe6, 0xe9, 0xd3, 0xa7, 0x52, 0x00, 0x4a, 0xdd, 0x3c, 0xff,
00731 0x49, 0x14, 0xf3, 0xe6, 0xcd, 0xa3, 0x61, 0xc3, 0x86, 0xec, 0xde, 0xbd,
00732 0x5b, 0xad, 0x5d, 0xa5, 0x4a, 0x95, 0xd8, 0xbc, 0x79, 0x33, 0x27, 0x4e,
00733 0x9c, 0xa0, 0x63, 0xc7, 0x8e, 0x98, 0x98, 0x98, 0x60, 0x6c, 0x6c, 0xcc,
00734 0xdb, 0x6d, 0xde, 0xe6, 0xf0, 0xe1, 0xc3, 0xec, 0xfc, 0x65, 0x27, 0xb6,
00735 0xb6, 0xea, 0xe7, 0xfd, 0x47, 0xc2, 0x1f, 0xf3, 0x71, 0xd0, 0x4d, 0x36,
00736 0x38, 0x18, 0x91, 0xe8, 0x6a, 0x2f, 0x23, 0xb1, 0xa4, 0xe7, 0xf9, 0x16,
00737 0x26, 0x5c, 0xab, 0x5a, 0x8e, 0x61, 0x49, 0xb1, 0x7c, 0x7b, 0xf7, 0x0e,
00738 0xf9, 0x79, 0x79, 0x6a, 0x6d, 0x47, 0x8c, 0x1c, 0x41, 0x70, 0x70, 0x30,
00739 0x5f, 0x7d, 0xf5, 0x15, 0x9d, 0x3b, 0x77, 0xe6, 0xc4, 0x89, 0x13, 0xac,
00740 0x5e, 0xbd, 0x1a, 0x43, 0x43, 0xf5, 0xee, 0x3f, 0x65, 0xd2, 0x14, 0xda,
00741 0xb4, 0x69, 0xc3, 0x81, 0x03, 0x07, 0xc8, 0xc9, 0xc9, 0x91, 0x02, 0xa0,
00742 0xeb, 0xa4, 0xa6, 0xa6, 0xb2, 0x7e, 0xfd, 0x7a, 0x3c, 0xeb, 0xd4, 0x65,
00743 0xe6, 0xcc, 0x99, 0x8a, 0xb6, 0xb3, 0x66, 0xcd, 0xe2, 0xf2, 0xe5, 0xcb,
00744 0x0c, 0x1e, 0x3c, 0x18, 0x3b, 0xbb, 0x7f, 0x6e, 0x0f, 0x59, 0x59, 0x59,
00745 0xd1, 0xc7, 0xa7, 0x0f, 0xf7, 0xee, 0xdd, 0x63, 0xce, 0x9c, 0x39, 0x8a,
00746 0x9f, 0xb5, 0xfa, 0xc1, 0x7d, 0xfa, 0xc7, 0x47, 0x71, 0xbc, 0x8a, 0x03,
00747 0xd9, 0xce, 0x36, 0x32, 0x32, 0x5f, 0xf5, 0xe8, 0xce, 0xc4, 0x88, 0x30,
00748 0x57, 0x1b, 0xe6, 0x98, 0xe6, 0x31, 0xdc, 0xff, 0x3a, 0x77, 0x62, 0x63,
00749 0xd4, 0xda, 0x76, 0xe8, 0xd0, 0x81, 0x0b, 0x17, 0x2e, 0xb0, 0x72, 0xc5,
00750 0x4a, 0x6a, 0xd7, 0xae, 0xfd, 0x7f, 0xff, 0x6e, 0x6f, 0xcf, 0xc8,
00751 0x91, 0x23, 0x09, 0x0b, 0x0b, 0x63, 0xf4, 0xe8, 0xd1, 0x6a, 0xdf, 0x1f,
00752 0x12, 0x12, 0x42, 0x8f, 0x1e, 0x3d, 0xe8, 0xdb, 0xaf, 0x2f, 0x37, 0xfd,
```

```
00753 0x6f, 0x96, 0xc9, 0x69, 0x41, 0xa9, 0x17, 0x80, 0x9c, 0x9c, 0x1c, 0xfe,
00754 0xf8, 0xe3, 0x0f, 0x3a, 0x74, 0xe8, 0xc0, 0xb0, 0x61, 0xc3, 0xc8, 0x50,
00755 0xb3, 0xe2, 0x0b, 0x30, 0x78, 0xf0, 0x60, 0x02, 0x02, 0x03, 0x98, 0x35,
00756 0x6b, 0x16, 0x95, 0x2a, 0x55, 0xd2, 0xf8, 0xd9, 0x6e, 0x6e, 0x6e, 0xcc,
00757 0x98, 0x31, 0x83, 0xc0, 0x5b, 0x81, 0xf4, 0xe9, 0xd3, 0x47, 0xfd, 0xfc,
00758 0x33, 0x21, 0x81, 0xc9, 0x81, 0xfe, 0x8c, 0xc9, 0x48, 0xc4, 0xbf, 0xb2,
00759 0x3d, 0x05, 0xa6, 0x46, 0x32, 0x52, 0x5f, 0x01, 0x09, 0x15, 0x1d, 0x58,
00760 0x67, 0x6b, 0x48, 0xaf, 0x7b, 0x41, 0x1c, 0x51, 0x98, 0xe7, 0x3b, 0x3b,
00761 0x3b, 0xf3, 0xcb, 0x2f, 0xbf, 0x70, 0xe0, 0xc0, 0x01, 0xde, 0x7a, 0xeb,
00762 0x2d, 0x8c, 0x8d, 0x8d, 0x8b, 0xb5, 0xab, 0x56, 0xad, 0x1a, 0xbe, 0xbe,
00763 0xbe, 0x9c, 0x3e, 0x73, 0x5a, 0x71, 0xda, 0xb7, 0x6f, 0xef, 0x3e, 0x9a,
00764 0x34, 0x6e, 0xc2, 0xdc, 0xb9, 0x73, 0x89, 0x51, 0x10, 0x1c, 0x29, 0x00,
00765 0x25, 0x3c, 0xcf, 0xbf, 0x79, 0xf3, 0x26, 0x83, 0x07, 0x0f, 0xa6, 0x73,
00766 0xe7, 0xce, 0x5c, 0xbf, 0x7e, 0x5d, 0xad, 0xed, 0x9b, 0x6f, 0xbe, 0xc9,
00767 0xc9, 0x93, 0x27, 0x59, 0xbf, 0x7e, 0x3d, 0x0d, 0xea, 0x37, 0xc0, 0xc0,
00768 0x40, 0xfb, 0x79, 0xbb, 0x81, 0x81, 0x01, 0xf5, 0xeb, 0xd5, 0xe7, 0xe7,
00769 0x9f, 0x7f, 0xe6, 0xcf, 0x13, 0x7f, 0xf2, 0xc6, 0x1b, 0x6f, 0xa8, 0xb5,
00770 0xfd, 0x2b, 0x26, 0x9a, 0xcf, 0x82, 0x02, 0xf8, 0xce, 0xb2, 0x90, 0x88,
00771 0xca, 0x0e, 0x32, 0x62, 0x5f, 0x12, 0x2a, 0x47, 0x2b, 0x8e, 0x55, 0x73,
00772 0x64, 0x60, 0xfc, 0x13, 0x7e, 0xd2, 0x30, 0xcf, 0x9f, 0x3b, 0x6f, 0x2e,
00773 0x81, 0x81, 0x81, 0xf8, 0xf8, 0x60, 0x69, 0x69, 0xa9, 0xf1, 0xb3,
00774 0x8d, 0x8c, 0x8c, 0x68, 0xfb, 0x76, 0x5b, 0x0e, 0x1e, 0x3c, 0xc8, 0xfa,
00775 0xf5, 0xeb, 0x31, 0x33, 0x33, 0x53, 0x1c, 0x39, 0x36, 0x68, 0xd0, 0x80,
00776 0xdd, 0x7e, 0xbb, 0x51, 0xa9, 0x54, 0x52, 0x00, 0x5e, 0xdb, 0x3c, 0xff,
00777 0xf1, 0x63, 0x66, 0xcd, 0x9a, 0x45, 0x93, 0x26, 0x4d, 0xd8, 0xb5, 0x6b,
00778 0x97, 0x5a, 0x3b, 0x33, 0x33, 0x33, 0xb6, 0xc6, 0xd9, 0xc2, 0xf1, 0xe3,
00779 0xc7, 0x69, 0xdf, 0xbe, 0xbd, 0xda, 0x27, 0x81, 0x36, 0x98, 0x98, 0x98,
00780 0xd0, 0xe1, 0x9d, 0x0e, 0x1c, 0x3f, 0x7e, 0x9c, 0x75, 0xeb, 0xd7, 0x61,
00781 0x6a, 0x6a, 0xaa, 0xd6, 0x76, 0xcf, 0xa3, 0x70, 0x7a, 0x04, 0xf9, 0xb3,
00782 0xd3, 0xd5, 0x8a, 0x34, 0x77, 0x27, 0x19, 0xc1, 0xff, 0x92, 0x42, 0x3b,
00783 0x0b, 0xae, 0x97, 0xb7, 0x62, 0x5c, 0x76, 0x32, 0x53, 0x6f, 0xde, 0x24,
00784 0x46, 0x61, 0x51, 0xee, 0xd3, 0xcf, 0x3e, 0x25, 0x28, 0x28, 0x88, 0x6f,
00785 0xa6, 0x7d, 0x83, 0xab, 0xab, 0xeb, 0x73, 0xff, 0x2d, 0x1b, 0x1b, 0x1b,
00786 0x3e, 0xff, 0xfc, 0x73, 0xee, 0xde, 0xbb, 0xcb, 0xb8, 0xf1, 0xe3, 0xd4,
00787 0xda, 0xc5, 0x3f, 0x8d, 0xc7, 0xa7, 0xb7, 0x0f, 0x9f, 0x7c, 0xf2, 0x09,
00788 0xd7, 0xaf, 0x5f, 0xa7, 0xb0, 0xb0, 0x50, 0x0a, 0x40, 0x49, 0x91, 0x92,
00789 0x92, 0xc2, 0xba, 0xf5, 0xeb, 0xf0, 0xf0, 0xf0, 0x60, 0xee, 0xdc, 0xb9,
00790 0x8a, 0xb6, 0x73, 0xe6, 0xcc, 0xe1, 0xe1, 0xc3, 0x87, 0x0c, 0x1a, 0x34,
00791 0x08, 0x1b, 0x9b, 0x97, 0x37, 0x37, 0xb3, 0xb3, 0x63, 0xe8, 0xe7,
00792 0x43, 0x09, 0x0d, 0x0d, 0x65, 0xf2, 0xe4, 0xc9, 0x8a, 0xb6, 0x4b, 0xee,
00793 0xdd, 0xa1, 0x57, 0x78, 0x28, 0x67, 0xdc, 0xed, 0xc9, 0xb1, 0x32, 0x95,
00794 0x11, 0xfd, 0x1c, 0x3c, 0xaa, 0x68, 0xcf, 0xdc, 0x82, 0x0c, 0x86, 0x86,
00795 0xdc, 0xe1, 0x5a, 0x74, 0xb4, 0x5a, 0xbb, 0xe6, 0xcd, 0x9b, 0x73, 0xf2,
00796 0xe4, 0x49, 0xd6, 0xfe, 0xb4, 0x16, 0x6f, 0x6f, 0xef, 0xe7, 0x1a, 0xdd,
00797 0x15, 0x3b, 0x2d, 0xa8, 0x5a, 0x8d, 0xef, 0x17, 0x7f, 0xcf, 0xb9, 0x73,
00798 0xe7, 0x68, 0xd5, 0xaa, 0x95, 0x5a, 0xbb, 0x23, 0x47, 0x8e, 0xd0, 0xac,
00799 0x59, 0x33, 0xa6, 0xcf, 0x9c, 0xce, 0x93, 0x27, 0x4f, 0xa4, 0x00, 0xbc,
00800 0x4a, 0x72, 0x73, 0x73, 0x39, 0x76, 0xec, 0x18, 0x1d, 0x3a, 0x74, 0x60,
00801 0xf8, 0xb0, 0xe1, 0x8a, 0xaa, 0xfb, 0xd9, 0xb0, 0x21, 0x04, 0xde, 0x0a,
00802 0x64, 0xc6, 0x8c, 0x19, 0xb8, 0xb9, 0xb9, 0xbd, 0xb2, 0xef, 0x54, 0xa5,
00803 0x4a, 0x15, 0x16, 0x2c, 0x58, 0xc0, 0x8d, 0x1b, 0x37, 0xf8, 0xa4, 0xf7,
00804 0x27, 0x6a, 0xed, 0xe2, 0xd2, 0xd3, 0xf9, 0xea, 0x56, 0x00, 0x53, 0x4c,
00805 0xf3, 0xb8, 0x55, 0xc9, 0x9a, 0x42, 0x73, 0x13, 0x19, 0xdd, 0x4a, 0x22,
00806 0x5f, 0xc9, 0x91, 0x1d, 0xd5, 0xec, 0xf9, 0xf0, 0x4e, 0x00, 0xbf, 0x45,
00807 0x3d, 0x56, 0x6b, 0xe7, 0xe8, 0xe8, 0xc8, 0xc6, 0x8d, 0x1b, 0xff, 0x6f,
00808 0x74, 0x67, 0x62, 0xf2, 0xf2, 0xda, 0xd5, 0xc8, 0xc8, 0x88, 0xd6, 0xad,
00809 0x5b, 0x73, 0xe4, 0xc8, 0x11, 0x36, 0x6f, 0xd9, 0xac, 0xb8, 0x5b, 0xb0,
00810 0x70, 0xfe, 0x42, 0xde, 0x78, 0xe3, 0x0d, 0x76, 0xef, 0xde, 0x4d, 0x96,
00811 0x2a, 0x4b, 0x0a, 0xc0, 0xcb, 0x9e, 0xe7, 0x5f, 0xbf, 0x7e, 0x9d, 0xbe,
00812 0x7d, 0xfb, 0xf2, 0xde, 0x7b, 0xef, 0x71, 0xe3, 0xc6, 0x0d, 0xb5, 0xb6,
00813 0xad, 0x5a, 0xb5, 0xe2, 0xf4, 0xe9, 0xd3, 0xac, 0xfd, 0xf1, 0x27, 0xea,
00814 0xd7, 0xab, 0xff, 0xc2, 0x4f, 0x02, 0xad, 0x1a, 0xcf, 0xd0, 0x90, 0xc6,
00815 0x8d, 0x1b, 0xb3, 0xed, 0xe7, 0x6d, 0x1c, 0x38, 0x70, 0x80, 0xba, 0x5e,
00816 0x75, 0xd5, 0xda, 0x9e, 0x09, 0x0f, 0x67, 0xd0, 0xed, 0xdb, 0xf8, 0xda,
00817 0x18, 0x10, 0x53, 0xc5, 0x51, 0x46, 0xfa, 0xff, 0x90, 0xe3, 0x64, 0xcd,
00818 0x89, 0xca, 0x36, 0x7c, 0xfa, 0xf8, 0x21, 0xdf, 0xd0, 0x0c, 0x50, 0xb4,
00819 0x9d, 0x3e, 0x63, 0x3a, 0xfe, 0xfe, 0xfe, 0x7c, 0xf6, 0xd9, 0x67, 0x8a,
00820 0x5b, 0xb5, 0x2f, 0x8a, 0xad, 0xad, 0x2d, 0x83, 0x07, 0x0d, 0xe6, 0xe1,
00821 0xc3, 0x87, 0x8c, 0x1f, 0x3f, 0x5e, 0xad, 0x5d, 0x74, 0x74, 0x34, 0x3e,
00822 0x3e, 0x3e, 0x74, 0xef, 0xd6, 0x9d, 0x2b, 0x57, 0xae, 0x94, 0xaa, 0x69,
00823 0x81, 0xce, 0x0a, 0x40, 0x64, 0x64, 0x24, 0xd3, 0xa7, 0x4f, 0xa7, 0x59,
00824 0xb3, 0x66, 0xec, 0xdd, 0xbb, 0x57, 0xad, 0x9d, 0xb3, 0xb3, 0x33, 0x5b,
00825 0xb7, 0x6e, 0xe5, 0xe8, 0xd1, 0xa3, 0xb4, 0x6d, 0xdb, 0xf6, 0x85, 0xe6,
00826 0xf9, 0xff, 0x16, 0x33, 0x33, 0x33, 0xba, 0x75, 0xeb, 0xc6, 0xf9, 0xb3,
00827 0xe7, 0xf1, 0x5d, 0xb9, 0x02, 0x03, 0xd4, 0x8b, 0xcf, 0xd6, 0xb0, 0x50,
00828 0x7a, 0x3d, 0xc0, 0x63, 0xaf, 0x9b, 0x0d, 0xe9, 0x4e, 0x96, 0x7a, 0x1f,
00829 0xf8, 0x85, 0xe6, 0x26, 0x04, 0xba, 0xda, 0xf0, 0x55, 0x7a, 0x3c, 0x13,
00830 0x83, 0x82, 0x78, 0x94, 0x9a, 0xa2, 0xd6, 0xf6, 0x93, 0x5e, 0x9f, 0xe0,
00831 0xef, 0xef, 0xcf, 0xb7, 0x73, 0xbe, 0xc5, 0xdd, 0xdd, 0xbd, 0xc4, 0xbe,
00832 0x63, 0x95, 0x2a, 0x55, 0x58, 0xb2, 0x64, 0x09, 0x17, 0x2f, 0x5e, 0xa4,
00833 0x53, 0xa7, 0x4e, 0x6a, 0xed, 0x4e, 0x9c, 0x38, 0x41, 0x8b, 0x16, 0x2d,
00834 0x98, 0x39, 0x73, 0x26, 0x8f, 0x15, 0x46, 0x2f, 0x52, 0x00, 0x14, 0x48,
00835 0x4e, 0x49, 0x66, 0xed, 0xba, 0xb5, 0xbc, 0xd9, 0xe2, 0x4d, 0x16, 0x2c,
00836 0x58, 0xa0, 0x68, 0xbb, 0x60, 0xc1, 0x02, 0x6e, 0xdd, 0xba, 0xc5, 0x80,
00837 0x01, 0x03, 0x5e, 0xea, 0x3c, 0xff, 0xdf, 0xe2, 0xe4, 0xc4, 0x98,
00838 0x2f, 0xbe, 0x24, 0x34, 0x2c, 0x94, 0xaf, 0x27, 0x4c, 0x50, 0x6b, 0x97,
00839 0x91, 0x9e, 0xce, 0xbc, 0xbb, 0x41, 0x8c, 0xce, 0xcd, 0xe0, 0x9c, 0x8b,
```



```
00840    0x05, 0x79, 0xb6, 0xfa, 0x59, 0x76, 0x1c, 0xed, 0xee, 0x84, 0xaf, 0x83,
00841    0x09, 0x83, 0xef, 0x05, 0x71, 0xf9, 0x69, 0x9c, 0x5a, 0x3b, 0x6f, 0x6f,
00842    0x6f, 0x0e, 0x1f, 0x3e, 0xcc, 0xb6, 0xad, 0xdb, 0x68, 0xd8, 0xb0, 0x61,
00843    0x89, 0x8c, 0xee, 0x8a, 0x1b, 0xed, 0xb5, 0x6c, 0xd9, 0x92, 0x7d, 0xfb,
00844    0xf6, 0xb1, 0x7d, 0xfb, 0x76, 0x9c, 0x1c, 0xd5, 0x2f, 0xee, 0xce, 0x9f,
00845    0x3f, 0x9f, 0x26, 0x8d, 0x9b, 0xb0, 0x75, 0xeb, 0x56, 0x32, 0x33, 0x33,
00846    0xa5, 0x00, 0x68, 0x35, 0x04, 0xcc, 0xc9, 0xe1, 0xe8, 0xd1, 0xa3, 0xb4,
00847    0x6b, 0xdb, 0x8e, 0x11, 0xc3, 0x47, 0x10, 0x13, 0xad, 0x7e, 0xbf, 0x75,
00848    0xe0, 0xc0, 0x81, 0xdc, 0xba, 0x75, 0x8b, 0x29, 0x53, 0xa6, 0x50, 0xa1,
00849    0x42, 0x05, 0x9d, 0x6b, 0xd4, 0x1a, 0x35, 0x6a, 0xb0, 0xe8, 0xbb, 0xef,
00850    0xb8, 0x74, 0xe9, 0x12, 0x9d, 0x3b, 0x77, 0x56, 0x6b, 0x17, 0x14, 0x19,
00851    0xc9, 0xd8, 0xd0, 0x7b, 0x4c, 0x17, 0x59, 0x84, 0x56, 0xb6, 0x03, 0x4b,
00852    0xfd, 0x58, 0x28, 0x4c, 0xf3, 0xb0, 0xc3, 0xcf, 0xd5, 0x9a, 0x1e, 0xc1,
00853    0x41, 0x6c, 0xbd, 0x77, 0x57, 0xad, 0x9d, 0xb9, 0xb9, 0x39, 0xab, 0x7e,
00854    0x5c, 0xc5, 0xd9, 0xb3, 0x67, 0xe9, 0xd2, 0xa5, 0x8b, 0xe2, 0x16, 0x5d,
00855    0x49, 0x61, 0x69, 0x69, 0x49, 0xbf, 0x7e, 0xfd, 0x08, 0x0c, 0x0a, 0x54,
00856    0x4c, 0x38, 0x8b, 0x8f, 0x8f, 0x67, 0xd0, 0xa0, 0x41, 0x74, 0xeb, 0xd6,
00857    0x8d, 0x0b, 0x17, 0x2f, 0x50, 0x50, 0x20, 0x05, 0x40, 0x89, 0x7e,
00858    0xfd, 0xfa, 0xd1, 0xa5, 0x4b, 0x17, 0x02, 0x03, 0x03, 0xd5, 0xda, 0xb4,
00859    0x69, 0xd3, 0x86, 0xd3, 0x67, 0x4e, 0xb3, 0x71, 0xe3, 0x46, 0xea, 0xd5,
00860    0xab, 0xf7, 0x5a, 0x9e, 0x04, 0xcf, 0xb3, 0x90, 0xd4, 0xa2, 0x45, 0x0b,
00861    0xf6, 0xee, 0xdd, 0xcb, 0xee, 0xdd, 0xbb, 0xa9, 0x5a, 0xbb, 0x9a, 0x5a,
00862    0xdb, 0xe3, 0x8f, 0x23, 0xe9, 0x1d, 0x14, 0xc8, 0x5a, 0x3b, 0x13, 0x9e,
00863    0xba, 0x97, 0xdd, 0xf5, 0x81, 0x3c, 0x1b, 0x73, 0xce, 0xb9, 0x59, 0x33,
00864    0x38, 0x22, 0x92, 0x85, 0xf7, 0x6e, 0x93, 0xa7, 0x50, 0x79, 0x37, 0x61,
00865    0xd2, 0x04, 0xee, 0xdc, 0xb9, 0xc3, 0xe8, 0x51, 0xa3, 0x71, 0x74, 0xd4,
00866    0xbd, 0xaa, 0x9a, 0xe8, 0x56, 0x91, 0x59, 0xb3, 0x66, 0x71, 0xed, 0xda,
00867    0x35, 0xde, 0x7f, 0xff, 0x7d, 0xb5, 0x76, 0xa7, 0x4e, 0x9d, 0xa2, 0x75,
00868    0xab, 0xd6, 0x7c, 0xfd, 0xf5, 0xd7, 0x84, 0x87, 0x87, 0x4b, 0x01, 0x50,
00869    0xc7, 0xc1, 0x83, 0x07, 0xd5, 0xfe, 0x5f, 0xb5, 0x6a, 0xd5, 0xd8, 0xb1,
00870    0x63, 0x07, 0x47, 0x8e, 0x1c, 0xa1, 0xed, 0xdb, 0xaf, 0x67, 0x9e, 0xff,
00871    0x22, 0x4f, 0x8c, 0x5e, 0xbd, 0x7a, 0x71, 0xe9, 0xcc, 0x45, 0x96, 0x2e,
00872    0x5d, 0xaa, 0x68, 0xfb, 0xd3, 0xbd, 0x3b, 0x7c, 0x1e, 0x1f, 0xc3, 0x41,
00873    0x17, 0x0b, 0x32, 0x1d, 0xcb, 0x50, 0xb5, 0xa1, 0xb5, 0x39, 0x21, 0xee,
00874    0x0e, 0x4c, 0x23, 0x8b, 0xb1, 0x77, 0x6f, 0x13, 0x9e, 0x9c, 0xac, 0xd6,
00875    0xf4, 0xfd, 0xf7, 0xdf, 0xe7, 0xf2, 0xe5, 0xcb, 0x2c, 0x5a, 0xb8, 0x88,
00876    0x6a, 0xd5, 0xaa, 0xe9, 0xf6, 0xf0, 0xd9, 0xd0, 0x90, 0x66, 0xcd, 0x9a,
00877    0xb1, 0x67, 0xcf, 0x1e, 0xf6, 0xec, 0xd9, 0x83, 0xbd, 0xbd, 0xfa, 0x9a,
00878    0x10, 0x5f, 0x5f, 0x5f, 0xea, 0xd4, 0xa9, 0xc3, 0x86, 0x8d, 0x1b, 0x48,
00879    0x4b, 0x4b, 0xd3, 0x9d, 0x07, 0xd5, 0x6b, 0xf8, 0x9b, 0x55, 0x81, 0x41,
00880    0xda, 0x1a, 0x7f, 0xf7, 0xdd, 0x77, 0xac, 0x58, 0xb1, 0x82, 0x16, 0x2d,
00881    0x5a, 0x28, 0x26, 0xdf, 0xe8, 0x7c, 0x0c, 0x58, 0x5b, 0xd3, 0xa2, 0x45,
00882    0x0b, 0x7c, 0x7c, 0x7c, 0xc8, 0xc9, 0xc9, 0xe1, 0xe6, 0xcd, 0x9b, 0xc5,
00883    0x0f, 0x8f, 0x33, 0x32, 0x38, 0x93, 0x98, 0xc0, 0x5f, 0xc6, 0x06, 0x54,
00884    0x74, 0x76, 0xa0, 0x02, 0x86, 0x18, 0xe6, 0xbd, 0xdc, 0xe1, 0x63, 0x6c,
00885    0x41, 0x1e, 0x07, 0xf2, 0xfe, 0x7b, 0x6e, 0xea, 0x69, 0x60, 0x4c, 0x1b,
00886    0xb3, 0x97, 0xbf, 0x8e, 0x12, 0xeb, 0x62, 0xcd, 0x56, 0x73, 0x98, 0x76,
00887    0x27, 0x88, 0x87, 0x69, 0xa9, 0x6a, 0xed, 0xea, 0xd4, 0xad, 0xc3, 0x9a,
00888    0xd5, 0x6b, 0x98, 0x3e, 0x7d, 0x3a, 0x55, 0xab, 0x56, 0xd5, 0xe9, 0xd1,
00889    0xdd, 0xff, 0x62, 0x62, 0x62, 0x82, 0xa7, 0xa7, 0x27, 0x83, 0x06, 0x0d,
00890    0xc2, 0xca, 0xda, 0x4a, 0xed, 0x29, 0x53, 0x05, 0x05, 0x1c, 0x3a,
00891    0x78, 0x88, 0x53, 0xa7, 0x4e, 0xa1, 0x52, 0xa9, 0x8a, 0xdb, 0xd5, 0x8a,
00892    0x04, 0x36, 0x97, 0xe4, 0x77, 0x7f, 0x1d, 0xad, 0xdc, 0x0e, 0x38, 0xa5,
00893    0xc9, 0xe8, 0xd3, 0x4f, 0x3f, 0x65, 0xfc, 0xf8, 0xf1, 0x78, 0x7b, 0x7b,
00894    0x97, 0xb9, 0xa1, 0x70, 0x7e, 0x7e, 0x3e, 0xe7, 0xcf, 0x9f, 0x67, 0xc6,
00895    0xac, 0x19, 0x5c, 0x3c, 0x7f, 0x51, 0xd1, 0xb6, 0x67, 0xed, 0x3a, 0xf4,
00896    0xb3, 0xb5, 0xc1, 0x23, 0x2c, 0xfe, 0xa5, 0x75, 0x56, 0x40, 0x6e, 0x16,
00897    0x9f, 0x66, 0xfd, 0x77, 0x56, 0x5d, 0x4f, 0x43, 0x33, 0xa6, 0xdb, 0xba,
00898    0xbe, 0xb4, 0xdf, 0x98, 0xe1, 0x64, 0xc1, 0x09, 0x13, 0x43, 0x56, 0x27,
00899    0xc4, 0x12, 0x9f, 0xa0, 0x7c, 0xda, 0xce, 0xb2, 0x1f, 0x96, 0xd1, 0xbf,
00900    0x5f, 0x7f, 0x9c, 0x9d, 0x9d, 0x4b, 0x7d, 0xdf, 0x16, 0x8a, 0x42, 0x02,
00901    0xfc, 0x03, 0x58, 0xb8, 0x70, 0x21, 0x7b, 0xf6, 0xec, 0x79, 0xde, 0xb7,
00902    0x9f, 0x07, 0xda, 0xe8, 0xb5, 0x00, 0xb4, 0x7f, 0xa7, 0x3d, 0xb3, 0x66,
00903    0xcd, 0xa2, 0x65, 0x8b, 0x96, 0xa5, 0x6a, 0xa8, 0xff, 0x6f, 0x48, 0x4b,
00904    0x4b, 0xe3, 0xc0, 0x81, 0x03, 0x4c, 0x9c, 0x38, 0x91, 0xb8, 0xb8, 0x38,
00905    0x45, 0xdb, 0x71, 0x75, 0xbd, 0xe8, 0x8a, 0x11, 0x4e, 0x31, 0x69, 0x3a,
00906    0x2d, 0x00, 0x05, 0xf6, 0xe6, 0x5c, 0xb7, 0x34, 0x65, 0x75, 0xe2, 0x53,
00907    0x6e, 0xc5, 0xc5, 0x2a, 0xda, 0x0e, 0x1f, 0x3e, 0x9c, 0xaf, 0xbe, 0xfa,
00908    0xea, 0xbf, 0x2a, 0xf5, 0xca, 0x0a, 0x2a, 0x95, 0x8a, 0xc3, 0x87, 0x0f,
00909    0x33, 0x76, 0xc2, 0x38, 0x62, 0x22, 0xa3, 0x75, 0x56, 0x00, 0x74, 0x6a,
00910    0x0a, 0x60, 0x62, 0x67, 0xca, 0xda, 0x55, 0x3f, 0xd1, 0xf6, 0xed, 0xb6,
00911    0x18, 0x19, 0x95, 0xfd, 0x8a, 0x3a, 0x33, 0x33, 0x33, 0x1a, 0x34, 0x68,
00912    0x40, 0xdf, 0x7e, 0x7d, 0xb1, 0xb7, 0xb3, 0xe7, 0xf4, 0xe9, 0xd3, 0x6a,
00913    0x6d, 0xaf, 0x24, 0xc4, 0xb3, 0x3f, 0x2b, 0x03, 0x37, 0x37, 0x67, 0x2a,
00914    0x19, 0x1b, 0x61, 0x9c, 0x93, 0xaf, 0x5b, 0x53, 0x00, 0x03, 0x03, 0xee,
00915    0xbb, 0xd9, 0xf3, 0xa3, 0x41, 0x1e, 0xdf, 0x07, 0xd1, 0x25, 0x2e, 0x53,
00916    0x7d, 0x55, 0x66, 0xfb, 0xf6, 0xed, 0xd9, 0xb4, 0x69, 0x13, 0x23, 0x47,
00917    0x8e, 0x2c, 0x13, 0x4f, 0x7d, 0x75, 0xd3, 0x02, 0x2f, 0x2f, 0xfa,
00918    0xf5, 0xed, 0x8f, 0x83, 0xbd, 0x3d, 0xa7, 0x4e, 0x9d, 0xd2, 0xe6, 0x6d,
00919    0x25, 0x3e, 0x05, 0xd0, 0x29, 0x01, 0x28, 0xcc, 0x29, 0x60, 0xeb, 0xd6,
00920    0xad, 0xe4, 0xe6, 0xe5, 0x52, 0xab, 0x56, 0xad, 0x62, 0x6b, 0xf5, 0xcb,
00921    0x22, 0x36, 0x36, 0x36, 0xb4, 0x69, 0xd3, 0x86, 0x0f, 0x3f, 0xfc, 0x90,
00922    0x94, 0x94, 0x14, 0x6e, 0xdf, 0xbe, 0x5d, 0xac, 0x5d, 0x6e, 0x5e, 0x1e,
00923    0x27, 0x12, 0xe3, 0xb9, 0x6b, 0x63, 0x4e, 0xe5, 0x8a, 0xe5, 0x71, 0x16,
00924    0x60, 0x90, 0x93, 0xf7, 0xda, 0x05, 0x20, 0xbe, 0x9c, 0x35, 0xbb, 0x9d,
00925    0xcc, 0x98, 0x70, 0xe7, 0x16, 0xc1, 0x0a, 0x87, 0x6b, 0x56, 0xad, 0x56,
00926    0x95, 0xd5, 0x3f, 0xae, 0x66, 0xe6, 0xcc, 0x99, 0xd4, 0xae, 0x5d, 0x5b,
```

```
00927 0x31, 0xc5, 0xb6, 0xac, 0x60, 0x6d, 0x6d, 0x4d, 0xb3, 0x66, 0xcd, 0xc8,
00928 0xcd, 0xcb, 0xe5, 0xd2, 0xc5, 0x4b, 0x52, 0x00, 0x80, 0x7c, 0xa0, 0x39,
00929 0xa0, 0x36, 0x95, 0xeb, 0xc2, 0xf9, 0x0b, 0xf8, 0xf9, 0xf9, 0x61, 0x6b,
00930 0x67, 0x47, 0xf5, 0xea, 0xd5, 0x31, 0x37, 0x2f, 0xfb, 0x89, 0x32, 0x06,
00931 0x06, 0x06, 0x54, 0xa8, 0x50, 0x81, 0x6e, 0xdd, 0xba, 0xd1, 0xa2, 0x45,
00932 0x0b, 0xc2, 0xc2, 0xc2, 0x88, 0x56, 0x53, 0x04, 0x13, 0x95, 0x96, 0xc6,
00933 0x6f, 0xb1, 0xd1, 0x64, 0xb9, 0x96, 0xa3, 0x92, 0x9d, 0x05, 0x76, 0xe9,
00934 0xb9, 0xaf, 0x45, 0x00, 0xb2, 0x5d, 0xac, 0xf8, 0xd3, 0xd2, 0x98, 0x49,
00935 0x4f, 0xa3, 0x38, 0xfd, 0x58, 0x39, 0xf3, 0x6d, 0xde, 0xbc, 0x79, 0xac,
00936 0x5c, 0xb9, 0x92, 0x96, 0x2d, 0x5a, 0xea, 0xc4, 0x7e, 0x7e, 0x49, 0x20,
00937 0x84, 0xe0, 0xc6, 0x8d, 0x1b, 0x8c, 0x1d, 0x3b, 0x96, 0x4d, 0x9b, 0x36,
00938 0x69, 0x32, 0x4f, 0x01, 0x96, 0x02, 0x37, 0xcb, 0xfa, 0x1a, 0x00, 0x14,
00939 0x6d, 0x3f, 0xf6, 0x01, 0x16, 0x02, 0x95, 0x95, 0x0c, 0x9b, 0x34, 0x69,
00940 0xc2, 0x82, 0x05, 0x0b, 0x68, 0xd7, 0xae, 0xdd, 0x4b, 0x2d, 0xf8, 0x28,
00941 0x0d, 0xeb, 0x03, 0x7e, 0x7e, 0x8c, 0x1c, 0x39, 0x92, 0xfc, 0xfc,
00942 0x7c, 0x85, 0x69, 0x84, 0x39, 0x5f, 0x56, 0xae, 0x42, 0x97, 0x42, 0x63,
00943 0xec, 0x93, 0xb4, 0xcb, 0x3a, 0x7b, 0xd1, 0x35, 0x80, 0x02, 0x73, 0x13,
00944 0x82, 0xdc, 0x1c, 0x58, 0xf7, 0x34, 0x86, 0xcb, 0x1a, 0x52, 0x5e, 0x07,
00945 0x0c, 0x18, 0xc0, 0x84, 0x09, 0x13, 0xa8, 0x5f, 0xbf, 0x3e, 0xfa, 0x44,
00946 0x44, 0x64, 0x04, 0x6b, 0x56, 0xaf, 0x61, 0xd1, 0xa2, 0x45, 0x9a, 0x4c,
00947 0x0b, 0xff, 0x7e, 0xea, 0x4f, 0x03, 0x4a, 0xfc, 0x10, 0xc2, 0xd7, 0x35,
00948 0xd1, 0x16, 0x40, 0x10, 0xf0, 0x13, 0x50, 0x00, 0xbc, 0x01, 0x14, 0x1b,
00949 0xdd, 0x31, 0x31, 0x31, 0x6c, 0xdf, 0xbe, 0x9d, 0x88, 0x88, 0x08, 0x6a,
00950 0xd6, 0xaa, 0x85, 0xb3, 0xb3, 0x73, 0xa9, 0xda, 0x22, 0x7a, 0x91, 0xf5,
00951 0x81, 0xc6, 0x8d, 0x1b, 0x33, 0x70, 0xd0, 0x40, 0x4c, 0x8c, 0x4d, 0xb8,
00952 0x74, 0xa9, 0xf8, 0xe1, 0x63, 0x41, 0x41, 0x3e, 0x97, 0x92, 0x12, 0x39,
00953 0x6f, 0x52, 0x88, 0x6b, 0x25, 0x57, 0xdc, 0x4c, 0x8c, 0x31, 0xca, 0xca,
00954 0x7d, 0x65, 0x23, 0x80, 0x47, 0xe5, 0x6d, 0xd8, 0x60, 0x6f, 0xc4, 0xbc,
00955 0x00, 0x7f, 0xa2, 0x14, 0xf6, 0xb3, 0x9b, 0x37, 0x6f, 0xc6, 0xa6, 0xcd,
00956 0x9b, 0x18, 0x33, 0x66, 0xcc, 0x2b, 0xad, 0xca, 0xd4, 0x35, 0x52, 0x52,
00957 0x52, 0xd8, 0xb2, 0x65, 0x0b, 0xbd, 0x3e, 0xe9, 0xc5, 0x89, 0x13, 0x27,
00958 0x34, 0x99, 0x5f, 0x00, 0x7a, 0x02, 0x6b, 0x81, 0xd7, 0x92, 0x33, 0xfc,
00959 0xba, 0x57, 0xda, 0xf2, 0x81, 0x33, 0xc0, 0x76, 0xa0, 0x22, 0xe0, 0xa9,
00960 0x6e, 0x54, 0x12, 0x18, 0x18, 0xc8, 0x9a, 0xd5, 0xab, 0x31, 0x35, 0x35,
00961 0xa5, 0x46, 0x8d, 0x1a, 0x3a, 0x91, 0xfb, 0x5f, 0x12, 0xd8, 0xdb, 0xdb,
00962 0xd3, 0xb1, 0x63, 0x47, 0x3a, 0x77, 0xee, 0x4c, 0x42, 0x42, 0x02, 0x21,
00963 0x21, 0x21, 0xc5, 0xda, 0x25, 0x67, 0xaa, 0x38, 0x16, 0x17, 0xc3, 0x23,
00964 0x07, 0x6b, 0xdc, 0xec, 0x6d, 0x70, 0x36, 0x2e, 0xc0, 0x20, 0xbb, 0xf0,
00965 0xa5, 0x09, 0x40, 0x4a, 0x79, 0x6b, 0xf6, 0x58, 0x1a, 0x30, 0x2e, 0xec,
00966 0x1e, 0x41, 0xb1, 0xea, 0x1f, 0x54, 0x16, 0x16, 0x16, 0xac, 0x5d, 0xbb,
00967 0x96, 0x85, 0x0b, 0x17, 0xe2, 0xe9, 0xe9, 0xa9, 0x17, 0x8b, 0xb9, 0x50,
00968 0x74, 0xd5, 0xdc, 0x9f, 0x7f, 0xfe, 0xc9, 0xe0, 0xc1, 0x83, 0x59, 0xbf,
00969 0x7e, 0x3d, 0xd9, 0xd9, 0xd9, 0x8a, 0x3a, 0x0a, 0x8c, 0x00, 0x26, 0x00,
00970 0xd1, 0x48, 0xfe, 0x8f, 0xb7, 0x01, 0xff, 0xbf, 0x47, 0x08, 0x6a, 0x5f,
00971 0x26, 0x26, 0x26, 0x62, 0xfb, 0xf6, 0xed, 0x22, 0x3d, 0x3d, 0x5d, 0xe8,
00972 0x13, 0x2a, 0x95, 0x4a, 0x1c, 0x38, 0x70, 0x40, 0x34, 0x68, 0xd0, 0x40,
00973 0x68, 0x6a, 0xa3, 0x41, 0x35, 0x6a, 0x8a, 0x23, 0xb5, 0xeb, 0x09, 0x7f,
00974 0x7b, 0x8f, 0x7f, 0xbc, 0x36, 0x5b, 0xba, 0xfc, 0xc3, 0xbe, 0xa7, 0xa1,
00975 0x59, 0xb1, 0xb6, 0x57, 0x3c, 0x3c, 0xc5, 0xf2, 0xc6, 0xcd, 0x84, 0x87,
00976 0xa3, 0xa3, 0xc6, 0xbf, 0x39, 0x79, 0xf2, 0x64, 0x11, 0x11, 0x11, 0x21,
00977 0xf4, 0x8d, 0x80, 0xc0, 0x00, 0xd1, 0xbf, 0x7f, 0x7f, 0x8d, 0xed, 0xf3,
00978 0xf7, 0x53, 0x7e, 0x26, 0x20, 0x2f, 0x95, 0xd0, 0xb0, 0x2e, 0xf1, 0x39,
00979 0x10, 0xa7, 0xa9, 0x41, 0xdb, 0xb6, 0x6d, 0x2b, 0xce, 0x9e, 0x3b, 0x2b,
00980 0xf2, 0xf3, 0xf3, 0xf5, 0xca, 0xe1, 0x12, 0x12, 0x12, 0xc4, 0xf2, 0xe5,
00981 0xcb, 0x85, 0x81, 0x81, 0x81, 0x62, 0xfb, 0x98, 0xdb, 0x5b, 0x88, 0x19,
00982 0x75, 0x3c, 0xc5, 0x85, 0xba, 0x0d, 0x9f, 0x5b, 0x00, 0x6e, 0x56, 0xac,
00983 0x2d, 0xb6, 0x79, 0x35, 0x14, 0x6f, 0x57, 0xaf, 0xac, 0xd1, 0xb1, 0x7b,
00984 0xf6, 0xec, 0x29, 0xae, 0x5d, 0xbb, 0x26, 0x0a, 0x0a, 0x0a, 0xf4, 0xaa,
00985 0x1f, 0x1e, 0x47, 0x3d, 0x16, 0xb3, 0x66, 0xcd, 0xd2, 0x26, 0xf0, 0x05,
00986 0xf0, 0x8b, 0xa6, 0xf5, 0x2e, 0xc9, 0x7f, 0x63, 0xf7, 0xf7, 0xaa, 0x68,
00987 0x8e, 0xa6, 0xc6, 0x1d, 0x3d, 0x7a, 0xb4, 0x08, 0x0e, 0x0e, 0xd6, 0xbb,
00988 0x27, 0x4f, 0x48, 0x48, 0x88, 0x18, 0x35, 0x6a, 0x94, 0x46, 0xe7, 0xf3,
00989 0x76, 0xaf, 0x28, 0xd6, 0xd4, 0x6b, 0x24, 0xae, 0xb9, 0xd6, 0xd0, 0x4a,
00990 0x00, 0x0e, 0xd7, 0xf0, 0x12, 0xfd, 0xab, 0x57, 0xd7, 0xf8, 0xb9, 0xf5,
00991 0xea, 0xd5, 0x13, 0x87, 0x0e, 0x1f, 0x12, 0x39, 0x39, 0x39, 0x7a, 0xd5,
00992 0xee, 0xe9, 0xe9, 0xe9, 0x62, 0xcb, 0x96, 0x2d, 0xc2, 0xd5, 0xd5, 0x55,
00993 0x9b, 0xc0, 0xbf, 0x0a, 0xb4, 0x92, 0xe1, 0xfc, 0xef, 0xa9, 0x05, 0x1c,
00994 0xd1, 0xd4, 0xd0, 0x46, 0x46, 0x46, 0x62, 0xf1, 0xe2, 0xc5, 0x22, 0xee,
00995 0x69, 0x9c, 0x5e, 0x39, 0x63, 0x7e, 0x7e, 0xbe, 0xb8, 0x70, 0xf1, 0x82,
00996 0xe8, 0xd4, 0xa9, 0x93, 0x46, 0x67, 0xec, 0x5a, 0xab, 0xa6, 0xd8, 0x55,
00997 0xcb, 0x5b, 0x6c, 0xb2, 0x77, 0x2b, 0x56, 0x00, 0xce, 0xd7, 0x69, 0x28,
00998 0x66, 0x34, 0x68, 0x24, 0x8c, 0x8c, 0x8c, 0x14, 0x3f, 0xc7, 0xca, 0xca,
00999 0x4a, 0xf8, 0xfa, 0xfa, 0x8a, 0xa4, 0xa4, 0x24, 0xbd, 0x6a, 0xeb, 0xbc,
01000 0xbc, 0x3c, 0x71, 0xe6, 0xcc, 0x19, 0xd1, 0xa6, 0x4d, 0x1b, 0x6d, 0x02,
01001 0x3f, 0x06, 0x18, 0xac, 0x03, 0xeb, 0x6c, 0x65, 0x86, 0x77, 0x81, 0xbb,
01002 0x9a, 0x1a, 0xbe, 0x7a, 0xf5, 0xea, 0x62, 0xef, 0xde, 0xbd, 0x42, 0xa5,
01003 0x52, 0xe9, 0x95, 0x73, 0x66, 0x66, 0x8a, 0x5d, 0xbb, 0x76, 0x09,
01004 0x37, 0x37, 0x37, 0x8d, 0xce, 0xd9, 0xd4, 0xf5, 0x9f, 0x23, 0x80, 0x5a,
01005 0x66, 0xe6, 0xc2, 0xb3, 0xb2, 0xe6, 0x27, 0xda, 0xe8, 0xd1, 0xa3, 0x45,
01006 0x58, 0x58, 0x98, 0x5e, 0xb5, 0x6d, 0x61, 0x61, 0xa1, 0xb8, 0x73, 0xe7,
01007 0x8e, 0x18, 0x31, 0x62, 0x84, 0x36, 0x81, 0x9f, 0x0d, 0x2c, 0x02, 0xe4,
01008 0x75, 0xd2, 0xaf, 0x00, 0x13, 0x60, 0x0c, 0x90, 0xa4, 0xa9, 0x23, 0xba,
01009 0x77, 0xef, 0x2e, 0x2e, 0x5f, 0xbe, 0x2c, 0x0a, 0x0b, 0x0b, 0xf5, 0xca,
01010 0x59, 0x63, 0x62, 0x62, 0xc4, 0xfc, 0xf9, 0xf3, 0xb5, 0x9d, 0x97, 0x6a,
01011 0xfd, 0xea, 0xd2, 0xa5, 0x8b, 0xb8, 0x78, 0xe9, 0xa2, 0xde, 0xb5, 0x67,
01012 0x6c, 0x6c, 0xac, 0x58, 0xb4, 0x68, 0x91, 0xb6, 0xed, 0x74, 0x00, 0xa8,
01013 0x21, 0xc3, 0xf4, 0xd5, 0x53, 0x0e, 0x58, 0xf3, 0x77, 0x0e, 0x81, 0x62,
```

```
01014    0xa7, 0x8c, 0x1b, 0x37, 0x4e, 0xdc, 0xbf, 0x7f, 0x5f, 0xef, 0x9e, 0x58,
01015    0x41, 0xb7, 0x83, 0xc4, 0x90, 0x21, 0x43, 0x5e, 0x38, 0xf0, 0xdd, 0xdd,
01016    0xdd, 0x85, 0xdf, 0xaf, 0x7e, 0x22, 0x2b, 0x2b, 0x4b, 0xef, 0x46, 0x54,
01017    0xbf, 0xfe, 0xfa, 0xab, 0x70, 0x77, 0x77, 0xd7, 0xa6, 0x9d, 0x6e, 0x03,
01018    0xef, 0xc8, 0xb0, 0x2c, 0x79, 0xea, 0x01, 0x27, 0x35, 0x75, 0x90, 0x9d,
01019    0xbd, 0x9d, 0x58, 0xb5, 0x6a, 0x95, 0x48, 0x4e, 0xd6, 0xbb, 0x39,
01020    0xeb, 0xa9, 0x53, 0xa7, 0x44, 0xcb, 0x96, 0x2d, 0xff, 0x55, 0xf0, 0x2f,
01021    0x5c, 0xb8, 0x50, 0xc4, 0xc6, 0xc6, 0xea, 0x55, 0x9b, 0x15, 0x14, 0x14,
01022    0x88, 0x4b, 0x97, 0x2f, 0x89, 0x2e, 0x5d, 0xba, 0x68, 0xd3, 0x46, 0x89,
01023    0xc0, 0x68, 0xc0, 0x58, 0x86, 0xe2, 0xeb, 0xdd, 0x36, 0xfc, 0x10, 0x78,
01024    0xa8, 0xa9, 0xc3, 0x1a, 0x35, 0x6a, 0x24, 0x8e, 0x1c, 0x39, 0xa2, 0x77,
01025    0xab, 0xd6, 0x69, 0x69, 0x69, 0x62, 0xe3, 0xa6, 0x8d, 0xc2, 0xd9, 0xd9,
01026    0x59, 0xab, 0xc0, 0x1f, 0x3e, 0x7c, 0xb8, 0xb8, 0x7b, 0xef, 0xae, 0xde,
01027    0x0d, 0xf7, 0x43, 0x43, 0x43, 0xc5, 0x98, 0x31, 0x63, 0xb4, 0x69, 0xa3,
01028    0xbc, 0xbf, 0x47, 0xa0, 0xf2, 0xda, 0x27, 0x1d, 0xc2, 0x1c, 0xf8, 0x06,
01029    0xc8, 0xd0, 0xd4, 0x81, 0x7d, 0xfa, 0xf4, 0x11, 0x7f, 0xfd, 0xf5, 0x97,
01030    0xde, 0xed, 0x5b, 0x47, 0x46, 0x46, 0x8a, 0xe9, 0x33, 0xa6, 0xab, 0x6d,
01031    0x97, 0x56, 0xad, 0x5a, 0x89, 0x33, 0x67, 0xce, 0x88, 0xbc, 0xbc, 0x3c,
01032    0xbd, 0x6a, 0x97, 0xc4, 0xc4, 0x44, 0xb1, 0x72, 0xe5, 0x4a, 0x8d, 0xbb,
01033    0x1f, 0x7f, 0xbf, 0xfe, 0xa4, 0x28, 0x63, 0x55, 0xa2, 0xa3, 0xb8, 0x01,
01034    0x3f, 0x53, 0x94, 0x66, 0xac, 0xd8, 0x99, 0xdf, 0x7c, 0xf3, 0x8d, 0x88,
01035    0x8c, 0x8c, 0xd4, 0xbb, 0xfc, 0x01, 0x7f, 0x7f, 0x7f, 0xe1, 0xe3, 0xe3,
01036    0xf3, 0x7f, 0xed, 0x60, 0x68, 0x68, 0x28, 0x7e, 0xfe, 0xf9, 0x67, 0x91,
01037    0x96, 0x96, 0xa6, 0x57, 0xed, 0x90, 0x9d, 0x9d, 0x2d, 0x0e, 0x1f, 0x3e,
01038    0x2c, 0x1a, 0x35, 0x6e, 0xa4, 0x4d, 0xe0, 0x87, 0x01, 0xdd, 0x64, 0x78,
01039    0x95, 0x1e, 0x9a, 0x03, 0x17, 0x35, 0x75, 0xac, 0xad, 0xad, 0xd8,
01040    0xb2, 0x65, 0x8b, 0x48, 0x4d, 0x4d, 0xd5, 0x2b, 0xe7, 0xcf, 0xc9, 0xc9,
01041    0x11, 0x47, 0x8e, 0x1c, 0x11, 0xf3, 0xe7, 0xcf, 0x17, 0x51, 0x4f, 0xa2,
01042    0xf4, 0x6e, 0x9e, 0x7f, 0xed, 0xda, 0x35, 0xd1, 0xf3, 0xe3, 0x9e, 0xda,
01043    0x04, 0x7e, 0x1a, 0x30, 0xf9, 0xef, 0x11, 0xa6, 0xa4, 0x14, 0xae, 0x0f,
01044    0x0c, 0xf8, 0x3b, 0x29, 0x43, 0xb1, 0xa3, 0xdf, 0x6a, 0xd5, 0x4a, 0x9c,
01045    0x38, 0x71, 0x42, 0xe4, 0xe6, 0xe6, 0x0a, 0x49, 0xd9, 0x25, 0x22, 0x22,
01046    0x42, 0x4c, 0x99, 0x32, 0x45, 0x9b, 0xc0, 0x7f, 0x56, 0xa6, 0x5b, 0x5e,
01047    0x86, 0x51, 0xe9, 0xc7, 0x1a, 0x98, 0x0f, 0x64, 0x69, 0xea, 0xf8, 0xc1,
01048    0x83, 0x07, 0x8b, 0x5b, 0xb7, 0x6e, 0xc9, 0x48, 0x29, 0x63, 0xa4, 0xa6,
01049    0xa6, 0x8a, 0x8d, 0x1b, 0x37, 0x0a, 0x73, 0x73, 0x6d, 0x82, 0xff,
01050    0x02, 0xd0, 0x44, 0x86, 0x4d, 0xd9, 0xa3, 0x2a, 0xe0, 0xa7, 0xed, 0x16,
01051    0x58, 0x4c, 0x4c, 0x8c, 0x8c, 0x9c, 0x32, 0x30, 0xd5, 0xf9, 0xe3, 0x8f,
01052    0x3f, 0xc4, 0x9b, 0x2d, 0xde, 0xd4, 0x26, 0xf0, 0x9f, 0x00, 0x7d, 0x79,
01053    0x7d, 0x07, 0xe5, 0x48, 0x4a, 0x88, 0xb6, 0x40, 0x80, 0x26, 0x87, 0xa8,
01054    0x56, 0xab, 0x9a, 0xd8, 0xed, 0xb7, 0x5b, 0x64, 0x64, 0x64, 0xc8, 0x48,
01055    0x2a, 0x85, 0xdc, 0xba, 0x75, 0x4b, 0xf4, 0x1b, 0xa8, 0x75, 0x99, 0xee,
01056    0x6c, 0xc0, 0x4a, 0x86, 0x86, 0xfe, 0x60, 0x0c, 0x8c, 0xa4, 0xe8, 0x18,
01057    0x26, 0x45, 0x07, 0x69, 0xf7, 0x4e, 0x3b, 0x71, 0xf6, 0xac, 0xfe, 0x95,
01058    0x1d, 0x8f, 0x97, 0x56, 0xa2, 0x03, 0xa3, 0xc5, 0xbc, 0xf9, 0xf3, 0x9e, 0xa7,
01059    0x4c, 0xd7, 0x5d, 0x86, 0x83, 0xfe, 0xe2, 0x00, 0x7c, 0x0f, 0xe4, 0x6a,
01060    0x72, 0x96, 0x51, 0xa3, 0x47, 0x89, 0xd0, 0xd0, 0x50, 0x19, 0x61, 0x3a,
01061    0x4a, 0x5a, 0x5a, 0x9a, 0xd8, 0xb1, 0x73, 0x87, 0xa8, 0x5c, 0xb9, 0xb2,
01062    0x36, 0x81, 0x1f, 0x40, 0x09, 0x9f, 0xc1, 0x2f, 0xd1, 0x6d, 0x6a, 0x03,
01063    0xbf, 0x6b, 0xf3, 0xd4, 0xf0, 0xf5, 0xf5, 0x15, 0x09, 0x09, 0x09, 0x32,
01064    0xe2, 0x74, 0x84, 0xfc, 0xfc, 0x7c, 0x71, 0xe6, 0xcc, 0x19, 0xd1, 0xba,
01065    0x75, 0x6b, 0x6d, 0xcb, 0x74, 0x3f, 0x47, 0x96, 0xe9, 0x4a, 0xd4, 0xd0,
01066    0x15, 0x08, 0xd6, 0xe4, 0x48, 0x55, 0x6b, 0x57, 0x13, 0xfb, 0xf6, 0xed,
01067    0xd3, 0xbb, 0x22, 0x19, 0x5d, 0x23, 0x38, 0x38, 0x58, 0x0c, 0x1f, 0x31,
01068    0x5c, 0x9b, 0xc0, 0xcf, 0x01, 0x96, 0x50, 0x74, 0xd0, 0x8c, 0x44, 0xa2,
01069    0x88, 0x29, 0x30, 0x8e, 0xa2, 0xb3, 0xda, 0x15, 0x1d, 0xeb, 0x83, 0x0f,
01070    0x3e, 0x10, 0x57, 0xaf, 0x5e, 0xd5, 0xbb, 0xb4, 0xe2, 0xd7, 0x4d, 0x5c,
01071    0x5c, 0x9c, 0x58, 0xee, 0xbb, 0x5c, 0xdb, 0x79, 0xfe, 0x11, 0xa0, 0xa6,
01072    0x74, 0x6b, 0xc9, 0xf3, 0xe2, 0x0c, 0xac, 0x43, 0x8b, 0xb4, 0xe2, 0x49,
01073    0x93, 0x26, 0x89, 0x47, 0x8f, 0x1e, 0xc9, 0xc8, 0x7c, 0xc5, 0x64, 0x65,
01074    0x65, 0x89, 0x5f, 0xf7, 0xfc, 0x2a, 0xaa, 0x55, 0xab, 0xa6, 0x6d, 0x99,
01075    0xee, 0x7b, 0xd2, 0x8d, 0x25, 0x2f, 0x4a, 0x63, 0x8a, 0x8e, 0x2f, 0x57,
01076    0x74, 0x38, 0x1b, 0x1b, 0x1b, 0xb1, 0x66, 0xcd, 0x1a, 0xbd, 0x3b, 0x2a,
01077    0xab, 0x24, 0x28, 0x28, 0x28, 0x10, 0x57, 0xae, 0x5c, 0x11, 0xef, 0xbf,
01078    0xff, 0xbe, 0x36, 0x81, 0x9f, 0x04, 0x7c, 0xf9, 0xf7, 0x48, 0x4e, 0x22,
01079    0x79, 0x29, 0x18, 0x00, 0x9f, 0x00, 0xe1, 0x9a, 0x1c, 0xb0, 0x61, 0xc3,
01080    0x86, 0xe2, 0xe8, 0xef, 0x47, 0xf5, 0xae, 0xec, 0xf8, 0x55, 0xf1, 0xe0,
01081    0xc1, 0x03, 0x31, 0x71, 0xca, 0x44, 0x6d, 0x02, 0x3f, 0x9f, 0xa2, 0xcb,
01082    0x66, 0x64, 0x99, 0xae, 0xe4, 0x95, 0x61, 0x0e, 0xcc, 0x40, 0x8b, 0xb2,
01083    0x63, 0x1f, 0x1f, 0x1f, 0x11, 0x10, 0x18, 0x20, 0x23, 0xf8, 0x5f, 0x92,
01084    0x94, 0x94, 0x24, 0x56, 0xac, 0x5c, 0x21, 0x0c, 0xd0, 0x0d, 0xb5, 0x09,
01085    0xfe, 0x93, 0x40, 0x43, 0xe9, 0x9e, 0x92, 0x92, 0xa2, 0x12, 0x45, 0x37,
01086    0x1a, 0x69, 0x74, 0xce, 0x59, 0xb3, 0x67, 0xe9, 0x65, 0xd9, 0xf1, 0x8b,
01087    0xa4, 0xef, 0x1e, 0x3e, 0x7c, 0x58, 0x78, 0x79, 0x69, 0x13, 0xf8,
01088    0x0f, 0x81, 0x8f, 0xa4, 0x3b, 0x4a, 0x5e, 0x17, 0x6f, 0x01, 0x97, 0x35,
01089    0x39, 0xaa, 0x8b, 0x8b, 0x8b, 0xd8, 0xfc, 0xf3, 0x66, 0xbd, 0xab, 0xb9,
01090    0x7f, 0x1e, 0x0a, 0x0b, 0x45, 0x40, 0x40, 0x40, 0xe8, 0xd5, 0xb7,
01091    0xb7, 0xb6, 0x65, 0xba, 0x53, 0x91, 0x65, 0xba, 0x12, 0x1d, 0xc0, 0x90,
01092    0xa2, 0x33, 0xe0, 0x9f, 0x68, 0x72, 0xdc, 0x96, 0x2d, 0x5b, 0x8a, 0xd3,
01093    0xa7, 0x4f, 0xeb, 0xdd, 0xa9, 0x3b, 0x9a, 0x88, 0x8c, 0x8c, 0x14, 0x53,
01094    0xa6, 0x6a, 0x5d, 0xa6, 0xbb, 0x95, 0xa2, 0xbb, 0x24, 0x25, 0x12, 0x9d,
01095    0xc2, 0x96, 0xa2, 0x2b, 0xcf, 0x55, 0x9a, 0x1c, 0x79, 0xe8, 0xd0, 0xa1,
01096    0xe2, 0xce, 0x9d, 0x3b, 0x7a, 0x77, 0xee, 0xde, 0xff, 0xf2, 0xac, 0x4c,
01097    0xd7, 0xda, 0xda, 0x5a, 0x9b, 0xe0, 0xbf, 0x04, 0xbc, 0x29, 0xdd, 0x4c,
01098    0xa2, 0xeb, 0xd4, 0x00, 0xf6, 0x69, 0xb3, 0x3e, 0xb0, 0x60, 0xc1, 0x02,
01099    0xbd, 0x3b, 0x79, 0x57, 0x88, 0xa2, 0x13, 0x8b, 0x4f, 0x9e, 0x3a, 0x29,
01100    0x5a, 0xb4, 0x68, 0xa1, 0x4d, 0xe0, 0x47, 0x01, 0xfd, 0x91, 0x65, 0xba,
```

```

01101    0x92, 0x52, 0xc6, 0x3b, 0xc0, 0x2d, 0x4d, 0x0e, 0xee, 0xe6, 0xe6, 0x26,
01102    0x76, 0xfb, 0xed, 0xd6, 0x9b, 0xb4, 0xe2, 0xa0, 0xdb, 0x41, 0x62, 0xe0,
01103    0xc0, 0x81, 0xda, 0x04, 0xbe, 0x8a, 0xa2, 0x83, 0x5c, 0xac, 0xa5, 0x2b,
01104    0x49, 0x4a, 0x2b, 0xcf, 0xca, 0x8e, 0x13, 0x34, 0x39, 0x7c, 0xa7, 0x4e,
01105    0x9d, 0xc4, 0xa5, 0xcb, 0x97, 0xca, 0x6c, 0xd9, 0xf1, 0x73, 0x96, 0xe9,
01106    0xfe, 0xa0, 0x78, 0x48, 0xf7, 0x91, 0x94, 0x15, 0x1c, 0x81, 0x55, 0x68,
01107    0x51, 0x76, 0x3c, 0x7a, 0xf4, 0xe8, 0x32, 0x55, 0x76, 0xfc, 0xec, 0xde,
01108    0x42, 0x2d, 0xef, 0x25, 0xf0, 0x07, 0xda, 0x49, 0x77, 0x91, 0x94, 0x55,
01109    0x3c, 0x81, 0x63, 0x9a, 0x02, 0xc1, 0xc0, 0xc0, 0x40, 0xf8, 0xfa, 0xfa,
01110    0x8a, 0xc4, 0xc4, 0xc4, 0x52, 0x3d, 0xcf, 0x3f, 0x77, 0xee, 0x9c, 0x78,
01111    0xf7, 0xdd, 0x77, 0xb5, 0x09, 0xfc, 0x04, 0x60, 0x38, 0xb2, 0x4c, 0x57,
01112    0xa2, 0x27, 0x74, 0x05, 0x42, 0x35, 0x05, 0x46, 0x83, 0x06, 0x0d, 0xc4,
01113    0xc1, 0x43, 0x07, 0x45, 0x76, 0x76, 0x76, 0xa9, 0x2b, 0xd3, 0x1d, 0x35,
01114    0x6a, 0x94, 0xb6, 0xb7, 0xe9, 0x2e, 0x07, 0xec, 0xa5, 0x4b, 0x48, 0xf4,
01115    0x0d, 0x33, 0x8a, 0xce, 0x9c, 0x4f, 0xd5, 0x14, 0x28, 0x3d, 0x7a, 0xf4,
01116    0x10, 0x57, 0xae, 0x5e, 0xd1, 0xf9, 0x6d, 0xc3, 0x84, 0xc4, 0x04, 0xb1,
01117    0x6c, 0xd9, 0xb2, 0xe7, 0x29, 0xd3, 0xad, 0x23, 0xdd, 0x40, 0xa2, 0xef,
01118    0xb8, 0x00, 0x1b, 0xd1, 0xe2, 0xb6, 0xe3, 0x89, 0x13, 0x27, 0xea, 0x64,
01119    0xd9, 0xb1, 0x4a, 0xa5, 0x12, 0xfb, 0xf6, 0xed, 0x13, 0x75, 0xeb, 0xd6,
01120    0xd5, 0x26, 0xf0, 0x43, 0x90, 0x65, 0xba, 0x12, 0xc9, 0x3f, 0x68, 0x04,
01121    0x9c, 0xd3, 0x14, 0x40, 0xc6, 0xc6, 0xc6, 0x62, 0xc3, 0x86, 0x0d, 0x22,
01122    0x35, 0xed, 0xf5, 0xdf, 0x66, 0xf4, 0xac, 0x4c, 0xf7, 0x83, 0x0f, 0x3e,
01123    0xd0, 0x26, 0xf0, 0x93, 0x81, 0xf1, 0xc8, 0x32, 0x5d, 0x89, 0x44, 0x91,
01124    0xbe, 0xc0, 0x63, 0x4d, 0x01, 0xd5, 0xac, 0x59, 0x33, 0xf1, 0xfb, 0xef,
01125    0xbf, 0xbf, 0xb6, 0xb2, 0xe3, 0xf0, 0x47, 0xe1, 0x62, 0xfc, 0xf8, 0xf1,
01126    0xda, 0xde, 0xa6, 0xb6, 0x8e, 0xa2, 0x03, 0x56, 0x24, 0x12, 0x89, 0x16,
01127    0x58, 0x02, 0x73, 0xd0, 0xf2, 0xb6, 0xe3, 0x80, 0x80, 0x80, 0x12, 0x5b,
01128    0x1f, 0x48, 0x4e, 0x4e, 0x16, 0xeb, 0xd6, 0xad, 0x13, 0x56, 0x56, 0x56,
01129    0xda, 0x04, 0xff, 0x19, 0xa0, 0x81, 0xec, 0x4e, 0x89, 0xe4, 0xdf, 0x51,
01130    0x09, 0xd8, 0xa1, 0xcd, 0xa2, 0xda, 0xdc, 0xb9, 0x73, 0x45, 0x74, 0x4c,
01131    0xf4, 0x2b, 0x2f, 0xd3, 0x6d, 0xda, 0xb4, 0xa9, 0x36, 0x81, 0xff, 0x08,
01132    0xe8, 0x2d, 0xbb, 0x4f, 0x22, 0x79, 0x39, 0xb4, 0x02, 0x6e, 0x6a, 0x0a,
01133    0xbc, 0x0a, 0x15, 0x2a, 0x88, 0x6d, 0xdb, 0xb6, 0x89, 0xb4, 0xf4, 0x97,
01134    0x57, 0x76, 0x5c, 0x58, 0x58, 0x28, 0xfc, 0xfd, 0xfd, 0x45, 0xaf, 0x5e,
01135    0xbd, 0xb4, 0x09, 0xfc, 0x74, 0x60, 0x3a, 0x60, 0x21, 0xbb, 0x4c, 0x22,
01136    0x79, 0xb9, 0x18, 0x01, 0x9f, 0x02, 0xb1, 0x9a, 0x02, 0xb1, 0x75, 0x9b,
01137    0xd6, 0x2f, 0xa5, 0xec, 0x38, 0x2a, 0x2a, 0x4a, 0x7c, 0xfb, 0xed, 0xb7,
01138    0xda, 0x96, 0xe9, 0x6e, 0x03, 0xdc, 0x64, 0x37, 0x49, 0x24, 0xaf, 0x16,
01139    0x5b, 0x60, 0x31, 0x45, 0x49, 0x34, 0x1a, 0xcb, 0x8e, 0x83, 0x83, 0x83,
01140    0x9f, 0x3b, 0xf0, 0xd3, 0xd2, 0xd2, 0xc4, 0x96, 0x2d, 0x5b, 0x84, 0x99,
01141    0x99, 0x99, 0x36, 0xc1, 0xff, 0x17, 0x45, 0x07, 0xa3, 0x48, 0x24, 0x92,
01142    0x12, 0xa4, 0x06, 0x70, 0x48, 0x9b, 0xf5, 0x81, 0xa5, 0x4b, 0x97, 0x8a,
01143    0x98, 0x58, 0xcd, 0xb7, 0x1d, 0xe7, 0xe5, 0xe5, 0x89, 0x13, 0x27, 0x4e,
01144    0x88, 0x96, 0x2d, 0x5b, 0x6a, 0x7b, 0xcb, 0xce, 0x20, 0x64, 0x99, 0xae,
01145    0x44, 0xf2, 0x5a, 0x79, 0x97, 0xa2, 0x33, 0xf0, 0x15, 0x03, 0xb6, 0x56,
01146    0x9d, 0x5a, 0xc2, 0xcf, 0xcf, 0x4f, 0x64, 0x66, 0x65, 0x16, 0x1b, 0xfc,
01147    0x77, 0xef, 0xdd, 0x15, 0x43, 0x87, 0x0e, 0xd5, 0xb6, 0x4c, 0x77, 0x1e,
01148    0xb2, 0x4c, 0x57, 0x22, 0xd1, 0x19, 0x4c, 0x81, 0x51, 0x40, 0xa2, 0xa6,
01149    0x00, 0x7e, 0xef, 0xbd, 0xf7, 0xc4, 0xa5, 0x4b, 0x97, 0xfe, 0xef, 0x36,
01150    0xa3, 0x98, 0x98, 0x18, 0xb1, 0x68, 0xd1, 0x22, 0x6d, 0xd3, 0x77, 0xf7,
01151    0x03, 0xd5, 0x65, 0x73, 0x4b, 0x24, 0xba, 0x89, 0x23, 0xb0, 0x9a, 0xa2,
01152    0xe4, 0x1b, 0x8d, 0x69, 0xc5, 0xdb, 0xb6, 0x6d, 0x13, 0xae, 0x6e, 0xae,
01153    0xda, 0xde, 0xb2, 0xf3, 0x8e, 0x6c, 0x5e, 0x89, 0xa4, 0x74, 0x50, 0x1f,
01154    0xf8, 0x53, 0xcb, 0xa7, 0xba, 0x36, 0x65, 0xba, 0xc6, 0xb2, 0x49, 0x25,
01155    0x92, 0xd2, 0x47, 0x0f, 0xe0, 0xfe, 0xbf, 0x08, 0xfc, 0x3c, 0x60, 0xc5,
01156    0xdf, 0x23, 0x0a, 0x89, 0x44, 0x52, 0x8a, 0x79, 0x56, 0x76, 0x9c, 0xa6,
01157    0x65, 0xf0, 0x1f, 0xa3, 0xe8, 0xe0, 0x12, 0x89, 0x44, 0x52, 0x86, 0x70,
01158    0x05, 0xb6, 0xa0, 0xbe, 0xec, 0x38, 0x14, 0xf8, 0x40, 0x36, 0x93, 0x44,
01159    0x52, 0xb6, 0x79, 0x83, 0xa2, 0x33, 0xf6, 0x9f, 0x05, 0x7e, 0x2a, 0x30,
01160    0xe1, 0xef, 0x91, 0x82, 0x44, 0x22, 0xd1, 0x13, 0xfa, 0x02, 0x2b, 0x81,
01161    0x0a, 0xb2, 0x29, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12,
01162    0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48,
01163    0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22,
01164    0x91, 0x48, 0x24, 0x12, 0x89, 0x44, 0x22, 0x91, 0x48, 0x74, 0x9f, 0xff,
01165    0x07, 0x99, 0x3f, 0x80, 0xa7, 0x39, 0xed, 0x2c, 0x27, 0x00, 0x00, 0x00,
01166    0x00, 0x49, 0x45, 0x4e, 0x44, 0xae, 0x42, 0x60, 0x82};
01167
01169 constexpr ui32 icon_png_len = 13773;
01170
01171 } // namespace lava

```

## 5.15 liblava/app/imgui.hpp File Reference

ImGui integration.

```

#include "liblava/block/render_pipeline.hpp"
#include "liblava/file.hpp"
#include "liblava/frame/input.hpp"
#include "liblava/resource/texture.hpp"

```

```
#include "liblava/util/layer.hpp"
```

## Classes

- struct [lava::imgui](#)  
*ImGui integration.*
- struct [lava::imgui::icon\\_font](#)  
*ImGui icon font settings.*
- struct [lava::imgui::font](#)  
*ImGui font settings.*
- struct [lava::imgui::config](#)  
*ImGui configuration.*

## Functions

- void [lava::setup\\_imgui\\_font](#) ([imgui::config](#) &config, [imgui::font::ref](#) font)  
*Set up ImGui font.*
- void [lava::setup\\_imgui\\_font\\_icons](#) ([imgui::font](#) &font, [string](#) filename, [ui16](#) min, [ui16](#) max)  
*Set up imgui font icons.*
- void [lava::imgui\\_left\\_spacing](#) ([ui32](#) top=1)  
*ImGui left spacing with top offset.*

## Variables

- constexpr const [r32](#) [lava::default\\_imgui\\_font\\_size](#) = 18.f  
*Default ImGui font size.*

### 5.15.1 Detailed Description

ImGui integration.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.15.2 Function Documentation

#### 5.15.2.1 [imgui\\_left\\_spacing\(\)](#)

```
void lava::imgui_left_spacing (  
    ui32 top = 1)
```

ImGui left spacing with top offset.

## Parameters

<i>top</i>	Top offset
------------	------------

**5.15.2.2 setup\_imgui\_font()**

```
void lava::setup_imgui_font (
    ImGui::config & config,
    ImGui::font::ref font)
```

Set up ImGui font.

## Parameters

<i>config</i>	ImGui configuration
<i>font</i>	ImGui font

**5.15.2.3 setup\_imgui\_font\_icons()**

```
void lava::setup_imgui_font_icons (
    ImGui::font & font,
    string filename,
    uil6 min,
    uil6 max)
```

Set up imgui font icons.

## Parameters

<i>font</i>	ImGui font
<i>filename</i>	Font icon file name
<i>min</i>	Min range
<i>max</i>	Max range

**5.16 imgui.hpp**

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/block/render_pipeline.hpp"
00011 #include "liblava/file.hpp"
00012 #include "liblava/frame/input.hpp"
00013 #include "liblava/resource/texture.hpp"
00014 #include "liblava/util/layer.hpp"
00015
00016 // fwd
00017 struct GLFWwindow;
00018 struct GLFWcursor;
00019 struct ImDrawData;
00020 struct ImGuiStyle;
00021
00022 namespace lava {
```

```

00023
00025 constexpr const r32 default_imgui_font_size = 18.f;
00026
00030 struct imgui {
00032     using ptr = imgui*;
00033
00037     explicit imgui() = default;
00038
00043     explicit imgui(GLFWwindow* window) {
00044         setup(window);
00045     }
00046
00050     ~imgui() {
00051         destroy();
00052     }
00053
00057     struct icon_font {
00059         data font_data;
00060
00062         ui16 range_begin = 0;
00063
00065         ui16 range_end = 0;
00066
00068         r32 size = default_imgui_font_size;
00069     };
00070
00074     struct font {
00076         using ref = font const&;
00077
00079         string file;
00080
00082         r32 size = 21.f;
00083
00085         string icon_file;
00086
00088         r32 icon_size = 21.f;
00089
00091         ui16 icon_range_begin = 0;
00092
00094         ui16 icon_range_end = 0;
00095     };
00096
00100     struct config {
00102         data font_data;
00103
00105         r32 font_size = default_imgui_font_size;
00106
00108         std::shared_ptr<ImGuiStyle> style;
00109
00111         icon_font icon;
00112
00114         std::filesystem::path ini_file_dir;
00115
00117         i32 flags = 0;
00118     };
00119
00125     void setup(GLFWwindow* window, config config);
00126
00131     void setup(GLFWwindow* win) {
00132         setup(win, config());
00133     }
00134
00141     bool create(render_pipeline::s_ptr pipeline, index max_frames);
00142
00150     bool create(device::ptr dev,
00151                 index frames,
00152                 VkPipelineCache pipeline_cache) {
00153         return create(render_pipeline::make(dev, pipeline_cache),
00154                       frames);
00155     }
00156
00165     bool create(device::ptr dev,
00166                 index frames,
00167                 VkRenderPass pass,
00168                 VkPipelineCache pipeline_cache = 0) {
00169         if (!create(dev, frames, pipeline_cache))
00170             return false;
00171
00172         return m_pipeline->create(pass);
00173     }
00174
00180     bool upload_fonts(texture::s_ptr texture);
00181
00185     void destroy();
00186
00191     bool ready() const {
00192         return m_initialized;

```

```

00193     }
00194
00199     render_pipeline::s_ptr get_pipeline() {
00200         return m_pipeline;
00201     }
00202
00204     using draw_func = std::function<void()>;
00205
00207     draw_func on_draw;
00208
00210     layer_list layers;
00211
00216     bool capture_mouse() const;
00217
00222     bool capture_keyboard() const;
00223
00228     void set_active(bool value = true) {
00229         m_active = value;
00230     }
00231
00236     bool activated() const {
00237         return m_active;
00238     }
00239
00243     void toggle() {
00244         m_active = !m_active;
00245     }
00246
00251     void set_ini_file(std::filesystem::path dir);
00252
00257     std::filesystem::path get_ini_file() const {
00258         return std::filesystem::path(m_ini_file);
00259     }
00260
00264     void convert_style_to_srgb();
00265
00270     input_callback const& get_input_callback() const {
00271         return m_callback;
00272     }
00273
00274 private:
00282     void handle_key_event(i32 key, i32 scancode, i32 action, i32 mods);
00283
00290     void handle_mouse_button_event(i32 button, i32 action, i32 mods);
00291
00297     void handle_scroll_event(r64 x_offset, r64 y_offset);
00298
00303     void prepare_draw_lists(ImDrawData* draw_data);
00304
00309     void render_draw_lists(VkCommandBuffer cmd_buf);
00310
00314     void invalidate_device_objects();
00315
00319     void update_mouse_pos_and_buttons();
00320
00324     void update_mouse_cursor();
00325
00329     void new_frame();
00330
00335     void render(VkCommandBuffer cmd_buf);
00336
00338     device::ptr m_device = nullptr;
00339
00340     // Initialized state
00341     bool m_initialized = false;
00342
00344     render_pipeline::s_ptr m_pipeline;
00345
00347     pipeline_layout::s_ptr m_layout;
00348
00350     size_t m_buffer_memory_alignment = 256;
00351
00353     index m_frame = 0;
00354
00356     index m_max_frames = 4;
00357
00359     buffer::s_list m_vertex_buffers;
00360
00362     buffer::s_list m_index_buffers;
00363
00365     descriptor::s_ptr m_descriptor;
00366
00368     descriptor::pool::s_ptr m_descriptor_pool;
00369
00371     VkDescriptorSet m_descriptor_set = VK_NULL_HANDLE;
00372
00374     GLFWwindow* m_window = nullptr;

```



```

00375
00377     bool m_mouse_just_pressed[5] = {false, false, false, false, false};
00378
00380     r64 m_current_time = 0.0;
00381
00383     std::vector<GLFWcursor*> m_mouse_cursors;
00384
00386     string m_ini_file;
00387
00389     bool m_active = true;
00390
00392     input_callback m_callback;
00393
00395     std::array<ui16, 3> m_icons_range;
00396 };
00397
00403 void setup_imgui_font(imgui::config& config,
00404                      imgui::font::ref font);
00405
00413 void setup_imgui_font_icons(imgui::font& font,
00414                             string filename,
00415                             ui16 min, ui16 max);
00416
00421 void imgui_left_spacing(ui32 top = 1);
00422
00423 } // namespace lava

```

## 5.17 liblava/asset.hpp File Reference

Asset module.

```

#include "liblava/asset/load_image.hpp"
#include "liblava/asset/load_mesh.hpp"
#include "liblava/asset/load_texture.hpp"
#include "liblava/asset/write_image.hpp"

```

### 5.17.1 Detailed Description

Asset module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.18 asset.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/asset/load_image.hpp"
00011 #include "liblava/asset/load_mesh.hpp"
00012 #include "liblava/asset/load_texture.hpp"
00013 #include "liblava/asset/write_image.hpp"

```

## 5.19 liblava/asset/load\_image.hpp File Reference

Load image data from file and memory.

```
#include "liblava/resource/image.hpp"
```

### Functions

- [image\\_data::s\\_ptr](#) [lava::load\\_image](#) ([string\\_ref](#) filename)  
*Load image data from file.*
- [image\\_data::s\\_ptr](#) [lava::load\\_image](#) ([c\\_data::ref](#) data)  
*Load image data from memory.*

### 5.19.1 Detailed Description

Load image data from file and memory.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.19.2 Function Documentation

#### 5.19.2.1 `load_image()` [1/2]

```
image_data::s_ptr lava::load_image (
    c\_data::ref data)
```

Load image data from memory.

#### Parameters

<i>data</i>	Memory data to load
-------------	---------------------

#### Returns

[image\\_data::s\\_ptr](#) Loaded image

#### 5.19.2.2 `load_image()` [2/2]

```
image_data::s_ptr lava::load_image (
    string\_ref filename)
```

Load image data from file.

## Parameters

<i>filename</i>	File to load
-----------------	--------------

## Returns

image\_data::s\_ptr Loaded image

## 5.20 load\_image.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/resource/image.hpp"
00011
00012 namespace lava {
00013
00019 image_data::s_ptr load_image(string_ref filename);
00020
00026 image_data::s_ptr load_image(c_data::ref data);
00027
00028 } // namespace lava

```

## 5.21 liblava/asset/load\_mesh.hpp File Reference

Load mesh from file.

```
#include "liblava/resource/mesh.hpp"
```

## Functions

- [mesh::s\\_ptr lava::load\\_mesh \(device::ptr device, string\\_ref filename, string\\_ref temp\\_dir\)](#)  
*Load mesh from file.*

### 5.21.1 Detailed Description

Load mesh from file.

## Authors

Lava Block OÜ and contributors

## Copyright

Copyright (c) 2018-present, MIT License

### 5.21.2 Function Documentation

#### 5.21.2.1 load\_mesh()

```

mesh::s_ptr lava::load_mesh (
    device::ptr device,
    string_ref filename,
    string_ref temp_dir)

```

Load mesh from file.

## Parameters

<i>device</i>	Vulkan device
<i>filename</i>	File to load
<i>temp_dir</i>	Temporary directory

## Returns

mesh::s\_ptr Loaded mesh

## 5.22 load\_mesh.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/resource/mesh.hpp"
00011
00012 namespace lava {
00013
00021 mesh::s_ptr load_mesh(device::ptr device,
00022                      string_ref filename,
00023                      string_ref temp_dir);
00024
00025 } // namespace lava
```

## 5.23 liblava/asset/load\_texture.hpp File Reference

Load texture from file.

```
#include "liblava/resource/texture.hpp"
```

## Functions

- [texture::s\\_ptr lava::load\\_texture](#) ([device::ptr device](#), [texture\\_file](#) tex\_file, [texture\\_type](#) type=texture\_type::tex↵\_2d)  
*Load texture from file.*
- [texture::s\\_ptr lava::load\\_texture](#) ([device::ptr device](#), [string\\_ref](#) filename, [VkFormat](#) format=VK\_FORMAT\_↵R8G8B8A8\_SRGB, [texture\\_type](#) type=texture\_type::tex\_2d)  
*Load texture from file with default format (sRGB)*
- [texture::s\\_ptr lava::create\\_default\\_texture](#) ([device::ptr device](#), [uv2](#) size={512, 512}, [v3](#) color=[v3](#)(1.f), [r32](#) alpha=0.7529f)  
*Create a default texture with checkerboard pattern.*

### 5.23.1 Detailed Description

Load texture from file.

## Authors

Lava Block OÜ and contributors

## Copyright

Copyright (c) 2018-present, MIT License

## 5.23.2 Function Documentation

### 5.23.2.1 create\_default\_texture()

```
texture::s_ptr lava::create_default_texture (
    device::ptr device,
    uv2 size = {512, 512},
    v3 color = v3(1.f),
    r32 alpha = 0.7529f)
```

Create a default texture with checkerboard pattern.

#### Parameters

<i>device</i>	Vulkan device
<i>size</i>	Size of texture
<i>color</i>	Color of texture
<i>alpha</i>	Alpha value of texture

#### Returns

texture::s\_ptr Loaded texture

### 5.23.2.2 load\_texture() [1/2]

```
texture::s_ptr lava::load_texture (
    device::ptr device,
    string_ref filename,
    VkFormat format = VK_FORMAT_R8G8B8A8_SRGB,
    texture_type type = texture_type::tex_2d) [inline]
```

Load texture from file with default format (sRGB)

#### Parameters

<i>device</i>	Vulkan device
<i>filename</i>	File to load
<i>format</i>	Format of texture
<i>type</i>	Type of texture

#### Returns

texture::s\_ptr Loaded texture

### 5.23.2.3 load\_texture() [2/2]

```
texture::s_ptr lava::load_texture (
    device::ptr device,
    texture_file tex_file,
    texture_type type = texture_type::tex_2d)
```

Load texture from file.

**Parameters**

<i>device</i>	Vulkan device
<i>tex_file</i>	Texture file
<i>type</i>	Type of texture

**Returns**

texture::s\_ptr Loaded texture

## 5.24 load\_texture.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/resource/texture.hpp"
00011
00012 namespace lava {
00013
00021 texture::s_ptr load_texture(device::ptr device,
00022                             texture_file tex_file,
00023                             texture_type type = texture_type::tex_2d);
00024
00033 inline texture::s_ptr load_texture(device::ptr device,
00034                                     string_ref filename,
00035                                     VkFormat format = VK_FORMAT_R8G8B8A8_SRGB,
00036                                     texture_type type = texture_type::tex_2d) {
00037     return load_texture(device, {filename, format}, type);
00038 }
00039
00048 texture::s_ptr create_default_texture(device::ptr device,
00049                                       uv2 size = {512, 512},
00050                                       v3 color = v3(1.f),
00051                                       r32 alpha = 0.7529f);
00052
00053 } // namespace lava

```

## 5.25 liblava/asset/write\_image.hpp File Reference

Write image data to file.

```
#include "liblava/resource/image.hpp"
```

**Functions**

- bool [lava::write\\_image\\_png](#) (device::ptr device, image::s\_ptr image, string\_ref filename, bool swizzle)  
*Write image data to png file.*

### 5.25.1 Detailed Description

Write image data to file.

**Authors**

Lava Block OÜ and contributors

**Copyright**

Copyright (c) 2018-present, MIT License

## 5.25.2 Function Documentation

### 5.25.2.1 write\_image\_png()

```
bool lava::write_image_png (
    device::ptr device,
    image::s_ptr image,
    string_ref filename,
    bool swizzle)
```

Write image data to png file.

#### Parameters

<i>device</i>	Vulkan device
<i>image</i>	Image to write
<i>filename</i>	File to write
<i>swizzle</i>	Swizzle data

#### Returns

Write was successful or failed

## 5.26 write\_image.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/resource/image.hpp"
00011
00012 namespace lava {
00013
00022 bool write_image_png(device::ptr device,
00023                     image::s_ptr image,
00024                     string_ref filename,
00025                     bool swizzle);
00026
00027 } // namespace lava
```

## 5.27 liblava/base.hpp File Reference

Base module.

```
#include "liblava/base/base.hpp"
#include "liblava/base/debug_utils.hpp"
#include "liblava/base/device.hpp"
#include "liblava/base/device_table.hpp"
#include "liblava/base/instance.hpp"
#include "liblava/base/memory.hpp"
#include "liblava/base/physical_device.hpp"
#include "liblava/base/platform.hpp"
#include "liblava/base/queue.hpp"
```

### 5.27.1 Detailed Description

Base module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.28 base.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/base.hpp"
00011 #include "liblava/base/debug_utils.hpp"
00012 #include "liblava/base/device.hpp"
00013 #include "liblava/base/device_table.hpp"
00014 #include "liblava/base/instance.hpp"
00015 #include "liblava/base/memory.hpp"
00016 #include "liblava/base/physical_device.hpp"
00017 #include "liblava/base/platform.hpp"
00018 #include "liblava/base/queue.hpp"
```

## 5.29 liblava/base/base.hpp File Reference

Vulkan base types.

```
#include "liblava/core/version.hpp"
#include "liblava/util/math.hpp"
#include "vulkan/vulkan.h"
#include "volk.h"
```

#### Classes

- struct [lava::vk\\_result](#)  
*Vulkan result.*
- struct [lava::target\\_callback](#)  
*Target callback.*



## Typedefs

- using **lava::VkVersion** = [ui32](#)  
*Vulkan version.*
- using **lava::VkObjectHandle** = [ui64](#)  
*Vulkan object handle.*
- using **lava::VkFormats** = std::vector<VkFormat>  
*List of Vulkan formats.*
- using **lava::VkImages** = std::vector<VkImage>  
*List of Vulkan images.*
- using **lava::VkImagesRef** = [VkImages](#) const&  
*Reference to list of Vulkan images.*
- using **lava::VkImageViews** = std::vector<VkImageView>  
*List of Vulkan images views.*
- using **lava::VkFramebuffers** = std::vector<VkFramebuffer>  
*List of Vulkan frame buffers.*
- using **lava::VkCommandPools** = std::vector<VkCommandPool>  
*List of Vulkan command pools.*
- using **lava::VkCommandBuffers** = std::vector<VkCommandBuffer>  
*List of Vulkan command buffers.*
- using **lava::VkFences** = std::vector<VkFence>  
*List of Vulkan fences.*
- using **lava::VkSemaphores** = std::vector<VkSemaphore>  
*List of Vulkan semaphores.*
- using **lava::VkPresentModeKHRs** = std::vector<VkPresentModeKHR>  
*List of Vulkan present modes.*
- using **lava::VkDescriptorSets** = std::vector<VkDescriptorSet>  
*List of Vulkan descriptor sets.*
- using **lava::VkDescriptorSetLayouts** = std::vector<VkDescriptorSetLayout>  
*List of Vulkan descriptor set layouts.*
- using **lava::VkDescriptorSetLayoutBindings** = std::vector<VkDescriptorSetLayoutBinding>  
*List of Vulkan descriptor set layout bindings.*
- using **lava::VkDescriptorPoolSizes** = std::vector<VkDescriptorPoolSize>  
*List of Vulkan descriptor pool sizes.*
- using **lava::VkDescriptorPoolSizesRef** = [VkDescriptorPoolSizes](#) const&  
*Reference to a list of Vulkan descriptor pool sizes.*
- using **lava::VkPushConstantRanges** = std::vector<VkPushConstantRange>  
*List of Vulkan push constant ranges.*
- using **lava::VkAttachmentReferences** = std::vector<VkAttachmentReference>  
*List of Vulkan attachment references.*
- using **lava::VkClearValues** = std::vector<VkClearValue>  
*List of Vulkan clear values.*
- using **lava::VkPipelineShaderStageCreateInfos** = std::vector<VkPipelineShaderStageCreateInfo>  
*List of Vulkan pipeline shader stage create infos.*
- using **lava::VkSpecializationMapEntries** = std::vector<VkSpecializationMapEntry>  
*List of Vulkan specialization map entries.*
- using **lava::VkVertexInputBindingDescriptions** = std::vector<VkVertexInputBindingDescription>  
*List of Vulkan vertex input binding descriptions.*
- using **lava::VkVertexInputAttributeDescriptions** = std::vector<VkVertexInputAttributeDescription>  
*List of Vulkan vertex input attribute descriptions.*

- using **lava::VkPipelineColorBlendAttachmentStates** = std::vector<VkPipelineColorBlendAttachmentState>  
*List of Vulkan pipeline color blend attachment states.*
- using **lava::VkPipelineStageFlagsList** = std::vector<VkPipelineStageFlags>  
*List of Vulkan pipeline stage flags.*
- using **lava::VkDynamicStates** = std::vector<VkDynamicState>  
*List of Vulkan dynamic states.*
- using **lava::VkQueueFamilyPropertiesList** = std::vector<VkQueueFamilyProperties>  
*List of Vulkan queue family properties.*
- using **lava::VkExtensionPropertiesList** = std::vector<VkExtensionProperties>  
*List of Vulkan extension properties.*
- using **lava::VkLayerPropertiesList** = std::vector<VkLayerProperties>  
*List of Vulkan layer properties.*
- using **lava::VkPhysicalDevices** = std::vector<VkPhysicalDevice>  
*List of Vulkan physical devices.*
- using **lava::VkAttachments** = std::vector<VkImageViews>  
*List of Vulkan attachments (image views)*
- using **lava::VkAttachmentsRef** = [VkAttachments](#) const&  
*Reference of Vulkan attachments (image views)*

## Enumerations

- enum class [lava::api\\_version](#) : index { **v1\_0** = 0 , **v1\_1** , **v1\_2** , **v1\_3** }  
*Vulkan API versions.*

## Functions

- bool [lava::check](#) (VkResult result)  
*Check a Vulkan result.*
- bool [lava::failed](#) (VkResult result)  
*Check if a Vulkan result failed.*
- string [lava::to\\_string](#) (VkResult result)  
*Convert a Vulkan result to string.*
- string [lava::vk\\_version\\_to\\_string](#) (VkVersion version)  
*Convert a Vulkan version to string.*
- sem\_version [lava::to\\_version](#) (VkVersion version)  
*Convert a Vulkan version to semantic version.*
- [VkVersion lava::to\\_vk\\_version](#) (sem\_version version)  
*Convert a semantic version to Vulkan version.*
- [api\\_version lava::to\\_api\\_version](#) (VkVersion version)  
*Convert a Vulkan version to API version.*

## Variables

- constexpr bool const **lava::build\_failed** = false  
*Build failed.*
- constexpr bool const **lava::build\_done** = true  
*Build done.*
- constexpr ui32 const **lava::Vk\_Limit\_DescriptorSets** = 4  
*Limit of Vulkan description sets.*
- constexpr ui32 const **lava::Vk\_Limit\_Bindings** = 16  
*Limit of Vulkan bindings.*
- constexpr ui32 const **lava::Vk\_Limit\_Attachments** = 8  
*Limit of Vulkan attachments.*
- constexpr ui32 const **lava::Vk\_Limit\_VertexAttribs** = 16  
*Limit of Vulkan vertex attributes.*
- constexpr ui32 const **lava::Vk\_Limit\_VertexBuffers** = 4  
*Limit of Vulkan vertex buffers.*
- constexpr ui32 const **lava::Vk\_Limit\_PushConstant\_Size** = 128  
*Limit of Vulkan push constant size.*
- constexpr ui32 const **lava::Vk\_Limit\_UBO\_Size** = 16 \* 1024  
*Limit of Vulkan UBO size.*

### 5.29.1 Detailed Description

Vulkan base types.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.29.2 Function Documentation

#### 5.29.2.1 check()

```
bool lava::check (
    VkResult result)
```

Check a Vulkan result.

#### Parameters

<i>result</i>	Result to check
---------------	-----------------

#### Returns

Okay or error

#### 5.29.2.2 failed()

```
bool lava::failed (
    VkResult result) [inline]
```

Check if a Vulkan result failed.

**Parameters**

<i>result</i>	Result to check
---------------	-----------------

**Returns**

Error or okay

**5.29.2.3 to\_api\_version()**

```
api_version lava::to_api_version (  
    VkVersion version)
```

Convert a Vulkan version to API version.

**Parameters**

<i>version</i>	Vulkan version to convert
----------------	---------------------------

**Returns**

api\_version Converted API version

**5.29.2.4 to\_string()**

```
string lava::to_string (  
    VkResult result)
```

Convert a Vulkan result to string.

**Parameters**

<i>result</i>	Result to convert
---------------	-------------------

**Returns**

string String of result

**5.29.2.5 to\_version()**

```
sem_version lava::to_version (  
    VkVersion version)
```

Convert a Vulkan version to semantic version.

**Parameters**

<i>version</i>	Vulkan version to convert
----------------	---------------------------

**Returns**

sem\_version Converted semantic version

**5.29.2.6 to\_vk\_version()**

```
VkVersion lava::to_vk_version (  
    sem_version version)
```

Convert a semantic version to Vulkan version.

**Parameters**

<i>version</i>	Semantic version to convert
----------------	-----------------------------

**Returns**

VkVersion Converted Vulkan version

**5.29.2.7 vk\_version\_to\_string()**

```
string lava::vk_version_to_string (  
    VkVersion version)
```

Convert a Vulkan version to string.

**Parameters**

<i>version</i>	Version to convert
----------------	--------------------

**Returns**

string String of version

## 5.30 base.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/version.hpp"
00011 #include "liblava/util/math.hpp"
00012
00013 // clang-format off
00014
00015 #define VK_NO_PROTOTYPES
00016 #include "vulkan/vulkan.h"
00017 #include "volk.h"
00018
00019 // clang-format on
00020
00021 namespace lava {
00022
00024 using VkVersion = ui32;
00025
00027 using VkObjectHandle = ui64;
00028
00030 using VkFormats = std::vector<VkFormat>;
00031
00033 using VkImages = std::vector<VkImage>;
00034
00036 using VkImagesRef = VkImages const&;
00037
00039 using VkImageViews = std::vector<VkImageView>;
00040
00042 using VkFramebuffers = std::vector<VkFramebuffer>;
00043
00045 using VkCommandPools = std::vector<VkCommandPool>;
00046
00048 using VkCommandBuffers = std::vector<VkCommandBuffer>;
00049
00051 using VkFences = std::vector<VkFence>;
00052
00054 using VkSemaphores = std::vector<VkSemaphore>;
00055
00057 using VkPresentModeKHRs = std::vector<VkPresentModeKHR>;
00058
00060 using VkDescriptorSets = std::vector<VkDescriptorSet>;
00061
00063 using VkDescriptorSetLayouts = std::vector<VkDescriptorSetLayout>;
00064
00066 using VkDescriptorSetLayoutBindings = std::vector<VkDescriptorSetLayoutBinding>;
00067
00069 using VkDescriptorPoolSizes = std::vector<VkDescriptorPoolSize>;
00070
00072 using VkDescriptorPoolSizesRef = VkDescriptorPoolSizes const&;
00073
00075 using VkPushConstantRanges = std::vector<VkPushConstantRange>;
00076
00078 using VkAttachmentReferences = std::vector<VkAttachmentReference>;
00079
00081 using VkClearValues = std::vector<VkClearValue>;
00082
00084 using VkPipelineShaderStageCreateInfos = std::vector<VkPipelineShaderStageCreateInfo>;
00085
00087 using VkSpecializationMapEntries = std::vector<VkSpecializationMapEntry>;
00088
00090 using VkVertexInputBindingDescriptions = std::vector<VkVertexInputBindingDescription>;
00091
00093 using VkVertexInputAttributeDescriptions = std::vector<VkVertexInputAttributeDescription>;
00094
00096 using VkPipelineColorBlendAttachmentStates = std::vector<VkPipelineColorBlendAttachmentState>;
00097
00099 using VkPipelineStageFlagsList = std::vector<VkPipelineStageFlags>;
00100
00102 using VkDynamicStates = std::vector<VkDynamicState>;
00103
00105 using VkQueueFamilyPropertiesList = std::vector<VkQueueFamilyProperties>;
00106
00108 using VkExtensionPropertiesList = std::vector<VkExtensionProperties>;
00109
00111 using VkLayerPropertiesList = std::vector<VkLayerProperties>;
00112
00114 using VkExtensionPropertiesList = std::vector<VkExtensionProperties>;
00115
00117 using VkPhysicalDevices = std::vector<VkPhysicalDevice>;
00118
00124 bool check(VkResult result);
00125

```

```

00131 inline bool failed(VkResult result) {
00132     return !check(result);
00133 }
00134
00140 string to_string(VkResult result);
00141
00147 string vk_version_to_string(VkVersion version);
00148
00154 sem_version to_version(VkVersion version);
00155
00161 VkVersion to_vk_version(sem_version version);
00162
00166 enum class api_version : index {
00167     vl_0 = 0,
00168     vl_1,
00169     vl_2,
00170     vl_3
00171 };
00172
00178 api_version to_api_version(VkVersion version);
00179
00183 struct vk_result {
00185     bool state = false;
00186
00188     VkResult value = VK_NOT_READY;
00189
00194     operator bool() {
00195         return state;
00196     }
00197 };
00198
00200 constexpr bool const build_failed = false;
00201
00203 constexpr bool const build_done = true;
00204
00206 using VkAttachments = std::vector<VkImageViews>;
00207
00209 using VkAttachmentsRef = VkAttachments const&;
00210
00214 struct target_callback {
00216     using c_ptr = target_callback const*;
00217
00219     using list = std::vector<target_callback*>;
00220
00222     using c_list = std::vector<c_ptr>;
00223
00225     using created_func = std::function<bool(VkAttachmentsRef, rect::ref)>;
00226
00228     created_func on_created;
00229
00231     using destroyed_func = std::function<void()>;
00232
00234     destroyed_func on_destroyed;
00235 };
00236
00238 constexpr ui32 const Vk_Limit_DescriptorSets = 4;
00239
00241 constexpr ui32 const Vk_Limit_Bindings = 16;
00242
00244 constexpr ui32 const Vk_Limit_Attachments = 8;
00245
00247 constexpr ui32 const Vk_Limit_VertexAttribs = 16;
00248
00250 constexpr ui32 const Vk_Limit_VertexBuffers = 4;
00251
00253 constexpr ui32 const Vk_Limit_PushConstant_Size = 128;
00254
00256 constexpr ui32 const Vk_Limit_UBO_Size = 16 * 1024;
00257
00258 } // namespace lava

```

## 5.31 liblava/base/debug\_utils.hpp File Reference

Debug utilities.

```

#include "liblava/base/base.hpp"
#include "liblava/core/def.hpp"

```

## Classes

- struct [lava::scoped\\_label< T >](#)  
*Scoped debug util label.*

## Macros

- `#define LAVA_DEBUG_UTILS LAVA_DEBUG`  
*Only active in debug - enable for release profiling.*

## Functions

- void [lava::begin\\_label](#) (VkCommandBuffer cmd\_buf, [name](#) label, [v4](#) color)
- void [lava::end\\_label](#) (VkCommandBuffer cmd\_buf)
- void [lava::insert\\_label](#) (VkCommandBuffer cmd\_buf, [name](#) label, [v4](#) color)
- void [lava::begin\\_label](#) (VkQueue [queue](#), [name](#) label, [v4](#) color)
- void [lava::end\\_label](#) (VkQueue [queue](#))
- void [lava::insert\\_label](#) (VkQueue [queue](#), [name](#) label, [v4](#) color)
- void [lava::set\\_object\\_name](#) (VkDevice [device](#), VkObjectType type, [VkObjectHandle](#) handle, [name](#) object)
- void [lava::set\\_object\\_tag](#) (VkDevice [device](#), VkObjectType type, [VkObjectHandle](#) handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_name](#) (VkDevice [device](#), [VkObjectHandle](#) handle, [name](#) object)  
*Set the object name.*
- void [lava::set\\_tag](#) (VkDevice [device](#), [VkObjectHandle](#) handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)  
*Set the object tag.*
- void [lava::set\\_instance\\_name](#) (VkDevice [device](#), VkInstance handle, [name](#) object)
- void [lava::set\\_instance\\_tag](#) (VkDevice [device](#), VkInstance handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_physical\\_device\\_name](#) (VkDevice [device](#), VkPhysicalDevice handle, [name](#) object)
- void [lava::set\\_physical\\_device\\_tag](#) (VkDevice [device](#), VkPhysicalDevice handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_device\\_name](#) (VkDevice [device](#), [name](#) object)
- void [lava::set\\_device\\_tag](#) (VkDevice [device](#), [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_queue\\_name](#) (VkDevice [device](#), VkQueue handle, [name](#) object)
- void [lava::set\\_queue\\_tag](#) (VkDevice [device](#), VkQueue handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_semaphore\\_name](#) (VkDevice [device](#), VkSemaphore handle, [name](#) object)
- void [lava::set\\_semaphore\\_tag](#) (VkDevice [device](#), VkSemaphore handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_command\\_buffer\\_name](#) (VkDevice [device](#), VkCommandBuffer handle, [name](#) object)
- void [lava::set\\_command\\_buffer\\_tag](#) (VkDevice [device](#), VkCommandBuffer handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_fence\\_name](#) (VkDevice [device](#), VkFence handle, [name](#) object)
- void [lava::set\\_fence\\_tag](#) (VkDevice [device](#), VkFence handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_device\\_memory\\_name](#) (VkDevice [device](#), VkDeviceMemory handle, [name](#) object)
- void [lava::set\\_device\\_memory\\_tag](#) (VkDevice [device](#), VkDeviceMemory handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_buffer\\_name](#) (VkDevice [device](#), VkBuffer handle, [name](#) object)
- void [lava::set\\_buffer\\_tag](#) (VkDevice [device](#), VkBuffer handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_image\\_name](#) (VkDevice [device](#), VkImage handle, [name](#) object)
- void [lava::set\\_image\\_tag](#) (VkDevice [device](#), VkImage handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_event\\_name](#) (VkDevice [device](#), VkEvent handle, [name](#) object)
- void [lava::set\\_event\\_tag](#) (VkDevice [device](#), VkEvent handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)
- void [lava::set\\_query\\_pool\\_name](#) (VkDevice [device](#), VkQueryPool handle, [name](#) object)
- void [lava::set\\_query\\_pool\\_tag](#) (VkDevice [device](#), VkQueryPool handle, [ui64](#) [name](#), [void\\_c\\_ptr](#) tag, [size\\_t](#) size)



- void `lava::set_buffer_view_name` (VkDevice `device`, VkBufferView handle, `name` object)
- void `lava::set_buffer_view_tag` (VkDevice `device`, VkBufferView handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_image_view_name` (VkDevice `device`, VkImageView handle, `name` object)
- void `lava::set_image_view_tag` (VkDevice `device`, VkImageView handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_shader_module_name` (VkDevice `device`, VkShaderModule handle, `name` object)
- void `lava::set_shader_module_tag` (VkDevice `device`, VkShaderModule handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_pipeline_cache_name` (VkDevice `device`, VkPipelineCache handle, `name` object)
- void `lava::set_pipeline_cache_tag` (VkDevice `device`, VkPipelineCache handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_pipeline_layout_name` (VkDevice `device`, VkPipelineLayout handle, `name` object)
- void `lava::set_pipeline_layout_tag` (VkDevice `device`, VkPipelineLayout handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_render_pass_name` (VkDevice `device`, VkRenderPass handle, `name` object)
- void `lava::set_render_pass_tag` (VkDevice `device`, VkRenderPass handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_pipeline_name` (VkDevice `device`, VkPipeline handle, `name` object)
- void `lava::set_pipeline_tag` (VkDevice `device`, VkPipeline handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_descriptor_set_layout_name` (VkDevice `device`, VkDescriptorSetLayout handle, `name` object)
- void `lava::set_descriptor_set_layout_tag` (VkDevice `device`, VkDescriptorSetLayout handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_sampler_name` (VkDevice `device`, VkSampler handle, `name` object)
- void `lava::set_sampler_tag` (VkDevice `device`, VkSampler handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_descriptor_pool_name` (VkDevice `device`, VkDescriptorPool handle, `name` object)
- void `lava::set_descriptor_pool_tag` (VkDevice `device`, VkDescriptorPool handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_descriptor_set_name` (VkDevice `device`, VkDescriptorSet handle, `name` object)
- void `lava::set_descriptor_set_tag` (VkDevice `device`, VkDescriptorSet handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_framebuffer_name` (VkDevice `device`, VkFramebuffer handle, `name` object)
- void `lava::set_framebuffer_tag` (VkDevice `device`, VkFramebuffer handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_command_pool_name` (VkDevice `device`, VkCommandPool handle, `name` object)
- void `lava::set_command_pool_tag` (VkDevice `device`, VkCommandPool handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_sampler_ycbcr_conversion_name` (VkDevice `device`, VkSamplerYcbcrConversion handle, `name` object)
- void `lava::set_sampler_ycbcr_conversion_tag` (VkDevice `device`, VkSamplerYcbcrConversion handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_descriptor_update_template_name` (VkDevice `device`, VkDescriptorUpdateTemplate handle, `name` object)
- void `lava::set_descriptor_update_template_tag` (VkDevice `device`, VkDescriptorUpdateTemplate handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_surface_name` (VkDevice `device`, VkSurfaceKHR handle, `name` object)
- void `lava::set_surface_tag` (VkDevice `device`, VkSurfaceKHR handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_swapchain_name` (VkDevice `device`, VkSwapchainKHR handle, `name` object)
- void `lava::set_swapchain_tag` (VkDevice `device`, VkSwapchainKHR handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_display_name` (VkDevice `device`, VkDisplayKHR handle, `name` object)
- void `lava::set_display_tag` (VkDevice `device`, VkDisplayKHR handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)
- void `lava::set_display_mode_name` (VkDevice `device`, VkDisplayModeKHR handle, `name` object)
- void `lava::set_display_mode_tag` (VkDevice `device`, VkDisplayModeKHR handle, `ui64 name`, `void_c_ptr tag`, `size_t size`)

- void `lava::set_debug_report_callback_name` (VkDevice `device`, VkDebugReportCallbackEXT handle, `name` object)
- void `lava::set_debug_report_callback_tag` (VkDevice `device`, VkDebugReportCallbackEXT handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_indirect_commands_layout_name` (VkDevice `device`, VkIndirectCommandsLayoutNV handle, `name` object)
- void `lava::set_indirect_commands_layout_tag` (VkDevice `device`, VkIndirectCommandsLayoutNV handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_debug_utils_messenger_name` (VkDevice `device`, VkDebugUtilsMessengerEXT handle, `name` object)
- void `lava::set_debug_utils_messenger_tag` (VkDevice `device`, VkDebugUtilsMessengerEXT handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_validation_cache_name` (VkDevice `device`, VkValidationCacheEXT handle, `name` object)
- void `lava::set_validation_cache_tag` (VkDevice `device`, VkValidationCacheEXT handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_acceleration_structure_nv_name` (VkDevice `device`, VkAccelerationStructureNV handle, `name` object)
- void `lava::set_acceleration_structure_nv_tag` (VkDevice `device`, VkAccelerationStructureNV handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_acceleration_structure_name` (VkDevice `device`, VkAccelerationStructureKHR handle, `name` object)
- void `lava::set_acceleration_structure_tag` (VkDevice `device`, VkAccelerationStructureKHR handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_performance_configuration_name` (VkDevice `device`, VkPerformanceConfigurationINTEL handle, `name` object)
- void `lava::set_performance_configuration_tag` (VkDevice `device`, VkPerformanceConfigurationINTEL handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_deferred_operation_name` (VkDevice `device`, VkDeferredOperationKHR handle, `name` object)
- void `lava::set_deferred_operation_tag` (VkDevice `device`, VkDeferredOperationKHR handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)
- void `lava::set_private_data_slot_name` (VkDevice `device`, VkPrivateDataSlotEXT handle, `name` object)
- void `lava::set_private_data_slot_tag` (VkDevice `device`, VkPrivateDataSlotEXT handle, `ui64 name`, `void_c_ptr` tag, `size_t` size)

### 5.31.1 Detailed Description

Debug utilities.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.31.2 Function Documentation

#### 5.31.2.1 `begin_label()` [1/2]

```
void lava::begin_label (
    VkCommandBuffer cmd_buf,
    name label,
    v4 color) [inline]
```

#### See also

`begin_label`

### 5.31.2.2 begin\_label() [2/2]

```
void lava::begin_label (
    VkQueue queue,
    name label,
    v4 color) [inline]
```

See also

begin\_label

### 5.31.2.3 end\_label() [1/2]

```
void lava::end_label (
    VkCommandBuffer cmd_buf) [inline]
```

See also

end\_label

### 5.31.2.4 end\_label() [2/2]

```
void lava::end_label (
    VkQueue queue) [inline]
```

See also

end\_label

### 5.31.2.5 insert\_label() [1/2]

```
void lava::insert_label (
    VkCommandBuffer cmd_buf,
    name label,
    v4 color) [inline]
```

See also

insert\_label

### 5.31.2.6 insert\_label() [2/2]

```
void lava::insert_label (
    VkQueue queue,
    name label,
    v4 color) [inline]
```

See also

insert\_label

#### 5.31.2.7 set\_acceleration\_structure\_name()

```
void lava::set_acceleration_structure_name (
    VkDevice device,
    VkAccelerationStructureKHR handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.8 set\_acceleration\_structure\_nv\_name()

```
void lava::set_acceleration_structure_nv_name (
    VkDevice device,
    VkAccelerationStructureNV handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.9 set\_acceleration\_structure\_nv\_tag()

```
void lava::set_acceleration_structure_nv_tag (
    VkDevice device,
    VkAccelerationStructureNV handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.10 set\_acceleration\_structure\_tag()

```
void lava::set_acceleration_structure_tag (
    VkDevice device,
    VkAccelerationStructureKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.11 set\_buffer\_name()

```
void lava::set_buffer_name (
    VkDevice device,
    VkBuffer handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.12 set\_buffer\_tag()

```
void lava::set_buffer_tag (
    VkDevice device,
    VkBuffer handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.13 set\_buffer\_view\_name()

```
void lava::set_buffer_view_name (
    VkDevice device,
    VkBufferView handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.14 set\_buffer\_view\_tag()

```
void lava::set_buffer_view_tag (
    VkDevice device,
    VkBufferView handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.15 set\_command\_buffer\_name()

```
void lava::set_command_buffer_name (
    VkDevice device,
    VkCommandBuffer handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.16 set\_command\_buffer\_tag()

```
void lava::set_command_buffer_tag (
    VkDevice device,
    VkCommandBuffer handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.17 set\_command\_pool\_name()

```
void lava::set_command_pool_name (
    VkDevice device,
    VkCommandPool handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.18 set\_command\_pool\_tag()

```
void lava::set_command_pool_tag (
    VkDevice device,
    VkCommandPool handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.19 set\_debug\_report\_callback\_name()

```
void lava::set_debug_report_callback_name (
    VkDevice device,
    VkDebugReportCallbackEXT handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.20 set\_debug\_report\_callback\_tag()

```
void lava::set_debug_report_callback_tag (
    VkDevice device,
    VkDebugReportCallbackEXT handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.21 set\_debug\_utils\_messenger\_name()

```
void lava::set_debug_utils_messenger_name (
    VkDevice device,
    VkDebugUtilsMessengerEXT handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.22 set\_debug\_utils\_messenger\_tag()

```
void lava::set_debug_utils_messenger_tag (
    VkDevice device,
    VkDebugUtilsMessengerEXT handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.23 set\_deferred\_operation\_name()

```
void lava::set_deferred_operation_name (
    VkDevice device,
    VkDeferredOperationKHR handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.24 set\_deferred\_operation\_tag()

```
void lava::set_deferred_operation_tag (
    VkDevice device,
    VkDeferredOperationKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.25 set\_descriptor\_pool\_name()

```
void lava::set_descriptor_pool_name (
    VkDevice device,
    VkDescriptorPool handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.26 set\_descriptor\_pool\_tag()

```
void lava::set_descriptor_pool_tag (
    VkDevice device,
    VkDescriptorPool handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()



### 5.31.2.27 set\_descriptor\_set\_layout\_name()

```
void lava::set_descriptor_set_layout_name (
    VkDevice device,
    VkDescriptorSetLayout handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.28 set\_descriptor\_set\_layout\_tag()

```
void lava::set_descriptor_set_layout_tag (
    VkDevice device,
    VkDescriptorSetLayout handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.29 set\_descriptor\_set\_name()

```
void lava::set_descriptor_set_name (
    VkDevice device,
    VkDescriptorSet handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.30 set\_descriptor\_set\_tag()

```
void lava::set_descriptor_set_tag (
    VkDevice device,
    VkDescriptorSet handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.31 set\_descriptor\_update\_template\_name()

```
void lava::set_descriptor_update_template_name (
    VkDevice device,
    VkDescriptorUpdateTemplate handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.32 set\_descriptor\_update\_template\_tag()

```
void lava::set_descriptor_update_template_tag (
    VkDevice device,
    VkDescriptorUpdateTemplate handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.33 set\_device\_memory\_name()

```
void lava::set_device_memory_name (
    VkDevice device,
    VkDeviceMemory handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.34 set\_device\_memory\_tag()

```
void lava::set_device_memory_tag (
    VkDevice device,
    VkDeviceMemory handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.35 set\_device\_name()

```
void lava::set_device_name (
    VkDevice device,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.36 set\_device\_tag()

```
void lava::set_device_tag (
    VkDevice device,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.37 set\_display\_mode\_name()

```
void lava::set_display_mode_name (
    VkDevice device,
    VkDisplayModeKHR handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.38 set\_display\_mode\_tag()

```
void lava::set_display_mode_tag (
    VkDevice device,
    VkDisplayModeKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.39 set\_display\_name()

```
void lava::set_display_name (
    VkDevice device,
    VkDisplayKHR handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.40 set\_display\_tag()

```
void lava::set_display_tag (
    VkDevice device,
    VkDisplayKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.41 set\_event\_name()

```
void lava::set_event_name (
    VkDevice device,
    VkEvent handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.42 set\_event\_tag()

```
void lava::set_event_tag (
    VkDevice device,
    VkEvent handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.43 set\_fence\_name()

```
void lava::set_fence_name (
    VkDevice device,
    VkFence handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.44 set\_fence\_tag()

```
void lava::set_fence_tag (
    VkDevice device,
    VkFence handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.45 set\_framebuffer\_name()

```
void lava::set_framebuffer_name (
    VkDevice device,
    VkFramebuffer handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.46 set\_framebuffer\_tag()

```
void lava::set_framebuffer_tag (
    VkDevice device,
    VkFramebuffer handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.47 set\_image\_name()

```
void lava::set_image_name (
    VkDevice device,
    VkImage handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.48 set\_image\_tag()

```
void lava::set_image_tag (
    VkDevice device,
    VkImage handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.49 set\_image\_view\_name()

```
void lava::set_image_view_name (
    VkDevice device,
    VkImageView handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.50 set\_image\_view\_tag()

```
void lava::set_image_view_tag (
    VkDevice device,
    VkImageView handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.51 set\_indirect\_commands\_layout\_name()

```
void lava::set_indirect_commands_layout_name (
    VkDevice device,
    VkIndirectCommandsLayoutNV handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.52 set\_indirect\_commands\_layout\_tag()

```
void lava::set_indirect_commands_layout_tag (
    VkDevice device,
    VkIndirectCommandsLayoutNV handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.53 set\_instance\_name()

```
void lava::set_instance_name (
    VkDevice device,
    VkInstance handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.54 set\_instance\_tag()

```
void lava::set_instance_tag (
    VkDevice device,
    VkInstance handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.55 set\_name()

```
void lava::set_name (
    VkDevice device,
    VkObjectHandle handle,
    name object) [inline]
```

Set the object name.

#### See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/VkObjectType.html>

#### Parameters

<i>device</i>	Vulkan device
<i>handle</i>	Object handle
<i>object</i>	Name of object

### 5.31.2.56 set\_object\_name()

```
void lava::set_object_name (
    VkDevice device,
    VkObjectType type,
    VkObjectHandle handle,
    name object) [inline]
```

#### See also

set\_object\_name

### 5.31.2.57 set\_object\_tag()

```
void lava::set_object_tag (
    VkDevice device,
    VkObjectType type,
    VkObjectHandle handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

#### See also

set\_object\_tag



#### 5.31.2.58 set\_performance\_configuration\_name()

```
void lava::set_performance_configuration_name (
    VkDevice device,
    VkPerformanceConfigurationINTEL handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.59 set\_performance\_configuration\_tag()

```
void lava::set_performance_configuration_tag (
    VkDevice device,
    VkPerformanceConfigurationINTEL handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.60 set\_physical\_device\_name()

```
void lava::set_physical_device_name (
    VkDevice device,
    VkPhysicalDevice handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.61 set\_physical\_device\_tag()

```
void lava::set_physical_device_tag (
    VkDevice device,
    VkPhysicalDevice handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.62 set\_pipeline\_cache\_name()

```
void lava::set_pipeline_cache_name (
    VkDevice device,
    VkPipelineCache handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.63 set\_pipeline\_cache\_tag()

```
void lava::set_pipeline_cache_tag (
    VkDevice device,
    VkPipelineCache handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.64 set\_pipeline\_layout\_name()

```
void lava::set_pipeline_layout_name (
    VkDevice device,
    VkPipelineLayout handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.65 set\_pipeline\_layout\_tag()

```
void lava::set_pipeline_layout_tag (
    VkDevice device,
    VkPipelineLayout handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.66 set\_pipeline\_name()

```
void lava::set_pipeline_name (
    VkDevice device,
    VkPipeline handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.67 set\_pipeline\_tag()

```
void lava::set_pipeline_tag (
    VkDevice device,
    VkPipeline handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.68 set\_private\_data\_slot\_name()

```
void lava::set_private_data_slot_name (
    VkDevice device,
    VkPrivateDataSlotEXT handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.69 set\_private\_data\_slot\_tag()

```
void lava::set_private_data_slot_tag (
    VkDevice device,
    VkPrivateDataSlotEXT handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.70 set\_query\_pool\_name()

```
void lava::set_query_pool_name (
    VkDevice device,
    VkQueryPool handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.71 set\_query\_pool\_tag()

```
void lava::set_query_pool_tag (
    VkDevice device,
    VkQueryPool handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.72 set\_queue\_name()

```
void lava::set_queue_name (
    VkDevice device,
    VkQueue handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.73 set\_queue\_tag()

```
void lava::set_queue_tag (
    VkDevice device,
    VkQueue handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.74 set\_render\_pass\_name()

```
void lava::set_render_pass_name (
    VkDevice device,
    VkRenderPass handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.75 set\_render\_pass\_tag()

```
void lava::set_render_pass_tag (
    VkDevice device,
    VkRenderPass handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.76 set\_sampler\_name()

```
void lava::set_sampler_name (
    VkDevice device,
    VkSampler handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.77 set\_sampler\_tag()

```
void lava::set_sampler_tag (
    VkDevice device,
    VkSampler handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.78 set\_sampler\_ycbcr\_conversion\_name()

```
void lava::set_sampler_ycbcr_conversion_name (
    VkDevice device,
    VkSamplerYcbcrConversion handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.79 set\_sampler\_ycbcr\_conversion\_tag()

```
void lava::set_sampler_ycbcr_conversion_tag (
    VkDevice device,
    VkSamplerYcbcrConversion handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.80 set\_semaphore\_name()

```
void lava::set_semaphore_name (
    VkDevice device,
    VkSemaphore handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.81 set\_semaphore\_tag()

```
void lava::set_semaphore_tag (
    VkDevice device,
    VkSemaphore handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.82 set\_shader\_module\_name()

```
void lava::set_shader_module_name (
    VkDevice device,
    VkShaderModule handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.83 set\_shader\_module\_tag()

```
void lava::set_shader_module_tag (
    VkDevice device,
    VkShaderModule handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

### 5.31.2.84 set\_surface\_name()

```
void lava::set_surface_name (
    VkDevice device,
    VkSurfaceKHR handle,
    name object) [inline]
```

See also

set\_name()

### 5.31.2.85 set\_surface\_tag()

```
void lava::set_surface_tag (
    VkDevice device,
    VkSurfaceKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.86 set\_swapchain\_name()

```
void lava::set_swapchain_name (
    VkDevice device,
    VkSwapchainKHR handle,
    name object) [inline]
```

See also

set\_name()

#### 5.31.2.87 set\_swapchain\_tag()

```
void lava::set_swapchain_tag (
    VkDevice device,
    VkSwapchainKHR handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

#### 5.31.2.88 set\_tag()

```
void lava::set_tag (
    VkDevice device,
    VkObjectHandle handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

Set the object tag.

See also

<https://www.khronos.org/registry/vulkan/specs/1.3-extensions/man/html/VkObjectType.html>

#### Parameters

<i>device</i>	Vulkan device
<i>handle</i>	Object handle
<i>name</i>	Name of tag
<i>tag</i>	Tag data
<i>size</i>	Size of tag data



**5.31.2.89 set\_validation\_cache\_name()**

```
void lava::set_validation_cache_name (
    VkDevice device,
    VkValidationCacheEXT handle,
    name object) [inline]
```

See also

set\_name()

**5.31.2.90 set\_validation\_cache\_tag()**

```
void lava::set_validation_cache_tag (
    VkDevice device,
    VkValidationCacheEXT handle,
    ui64 name,
    void_c_ptr tag,
    size_t size) [inline]
```

See also

set\_tag()

**5.32 debug\_utils.hpp**

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/base.hpp"
00011 #include "liblava/core/def.hpp"
00012
00014 #ifndef LAVA_DEBUG_UTILS
00015     #define LAVA_DEBUG_UTILS LAVA_DEBUG
00016 #endif
00017
00018 namespace lava {
00019
00020 #if LAVA_DEBUG_UTILS
00021
00028 void begin_label(VkCommandBuffer cmd_buf,
00029                 name label,
00030                 v4 color);
00031
00036 void end_label(VkCommandBuffer cmd_buf);
00037
00044 void insert_label(VkCommandBuffer cmd_buf,
00045                  name label,
00046                  v4 color);
00047
00054 void begin_label(VkQueue queue,
00055                  name label,
00056                  v4 color);
00057
00062 void end_label(VkQueue queue);
00063
00070 void insert_label(VkQueue queue,
00071                  name label,
00072                  v4 color);
00073
00081 void set_object_name(VkDevice device,
00082                     VkObjectType type,
00083                     VkObjectHandle handle,
00084                     name object);
00085
```

```

00095 void set_object_tag(VkDevice device,
00096                     VkObjectType type,
00097                     VkObjectHandle handle,
00098                     ui64 name,
00099                     void_c_ptr tag,
00100                     size_t size);
00101
00102 #else
00103
00105 inline void begin_label(VkCommandBuffer cmd_buf,
00106                       name label,
00107                       v4 color) {}
00108
00110 inline void end_label(VkCommandBuffer cmd_buf) {}
00111
00113 inline void insert_label(VkCommandBuffer cmd_buf,
00114                       name label,
00115                       v4 color) {}
00116
00118 inline void begin_label(VkQueue queue,
00119                       name label,
00120                       v4 color) {}
00121
00123 inline void end_label(VkQueue queue) {}
00124
00126 inline void insert_label(VkQueue queue,
00127                       name label,
00128                       v4 color) {}
00129
00131 inline void set_object_name(VkDevice device,
00132                           VkObjectType type,
00133                           VkObjectHandle handle,
00134                           name object) {}
00135
00137 inline void set_object_tag(VkDevice device,
00138                           VkObjectType type,
00139                           VkObjectHandle handle,
00140                           ui64 name,
00141                           void_c_ptr tag,
00142                           size_t size) {}
00143
00144 #endif
00145
00150 template <typename T>
00151 struct scoped_label {
00152     scoped_label(T scope, name label, v4 color = v4(0.f))
00153     : m_scope(scope) {
00154         begin_label(m_scope, label, color);
00155     }
00156
00157     ~scoped_label() {
00158         end_label(m_scope);
00159     }
00160
00161 private:
00162     T m_scope;
00163 };
00164
00182 inline void set_name(VkDevice device,
00183                   VkObjectHandle handle,
00184                   name object) {
00185     set_object_name(device,
00186                   VK_OBJECT_TYPE_UNKNOWN,
00187                   handle,
00188                   object);
00189 }
00190
00200 inline void set_tag(VkDevice device,
00201                   VkObjectHandle handle,
00202                   ui64 name,
00203                   void_c_ptr tag,
00204                   size_t size) {
00205     set_object_tag(device,
00206                   VK_OBJECT_TYPE_UNKNOWN,
00207                   handle,
00208                   name,
00209                   tag,
00210                   size);
00211 }
00212
00213 // ---
00214
00218 inline void set_instance_name(VkDevice device,
00219                             VkInstance handle,
00220                             name object) {
00221     set_object_name(device,
00222                   VK_OBJECT_TYPE_INSTANCE,

```

```

00223         VkObjectHandle(handle),
00224         object);
00225     }
00226
00230     inline void set_instance_tag(VkDevice device,
00231                                 VkInstance handle,
00232                                 ui64 name,
00233                                 void_c_ptr tag,
00234                                 size_t size) {
00235         set_object_tag(device,
00236                       VK_OBJECT_TYPE_INSTANCE,
00237                       VkObjectHandle(handle),
00238                       name,
00239                       tag,
00240                       size);
00241     }
00242
00243     // ---
00244
00248     inline void set_physical_device_name(VkDevice device,
00249                                          VkPhysicalDevice handle,
00250                                          name object) {
00251         set_object_name(device,
00252                       VK_OBJECT_TYPE_PHYSICAL_DEVICE,
00253                       VkObjectHandle(handle),
00254                       object);
00255     }
00256
00260     inline void set_physical_device_tag(VkDevice device,
00261                                       VkPhysicalDevice handle,
00262                                       ui64 name,
00263                                       void_c_ptr tag,
00264                                       size_t size) {
00265         set_object_tag(device,
00266                       VK_OBJECT_TYPE_PHYSICAL_DEVICE,
00267                       VkObjectHandle(handle),
00268                       name,
00269                       tag,
00270                       size);
00271     }
00272
00273     // ---
00274
00278     inline void set_device_name(VkDevice device,
00279                                 name object) {
00280         set_object_name(device,
00281                       VK_OBJECT_TYPE_DEVICE,
00282                       VkObjectHandle(device),
00283                       object);
00284     }
00285
00289     inline void set_device_tag(VkDevice device,
00290                               ui64 name,
00291                               void_c_ptr tag,
00292                               size_t size) {
00293         set_object_tag(device,
00294                       VK_OBJECT_TYPE_DEVICE,
00295                       VkObjectHandle(device),
00296                       name,
00297                       tag,
00298                       size);
00299     }
00300
00301     // ---
00302
00306     inline void set_queue_name(VkDevice device,
00307                               VkQueue handle,
00308                               name object) {
00309         set_object_name(device,
00310                       VK_OBJECT_TYPE_QUEUE,
00311                       VkObjectHandle(handle),
00312                       object);
00313     }
00314
00318     inline void set_queue_tag(VkDevice device,
00319                              VkQueue handle,
00320                              ui64 name,
00321                              void_c_ptr tag,
00322                              size_t size) {
00323         set_object_tag(device,
00324                       VK_OBJECT_TYPE_QUEUE,
00325                       VkObjectHandle(handle),
00326                       name,
00327                       tag,
00328                       size);
00329     }
00330

```

```

00331 // ---
00332
00336 inline void set_semaphore_name(VkDevice device,
00337                                VkSemaphore handle,
00338                                name object) {
00339     set_object_name(device,
00340                     VK_OBJECT_TYPE_SEMAPHORE,
00341                     VkObjectHandle(handle),
00342                     object);
00343 }
00344
00348 inline void set_semaphore_tag(VkDevice device,
00349                                VkSemaphore handle,
00350                                ui64 name,
00351                                void_c_ptr tag,
00352                                size_t size) {
00353     set_object_tag(device,
00354                    VK_OBJECT_TYPE_SEMAPHORE,
00355                    VkObjectHandle(handle),
00356                    name,
00357                    tag,
00358                    size);
00359 }
00360
00361 // ---
00362
00366 inline void set_command_buffer_name(VkDevice device,
00367                                      VkCommandBuffer handle,
00368                                      name object) {
00369     set_object_name(device,
00370                     VK_OBJECT_TYPE_COMMAND_BUFFER,
00371                     VkObjectHandle(handle),
00372                     object);
00373 }
00374
00378 inline void set_command_buffer_tag(VkDevice device,
00379                                    VkCommandBuffer handle,
00380                                    ui64 name,
00381                                    void_c_ptr tag,
00382                                    size_t size) {
00383     set_object_tag(device,
00384                    VK_OBJECT_TYPE_COMMAND_BUFFER,
00385                    VkObjectHandle(handle),
00386                    name,
00387                    tag,
00388                    size);
00389 }
00390
00391 // ---
00392
00396 inline void set_fence_name(VkDevice device,
00397                             VkFence handle,
00398                             name object) {
00399     set_object_name(device,
00400                     VK_OBJECT_TYPE_FENCE,
00401                     VkObjectHandle(handle),
00402                     object);
00403 }
00404
00408 inline void set_fence_tag(VkDevice device,
00409                            VkFence handle,
00410                            ui64 name,
00411                            void_c_ptr tag,
00412                            size_t size) {
00413     set_object_tag(device,
00414                    VK_OBJECT_TYPE_FENCE,
00415                    VkObjectHandle(handle),
00416                    name,
00417                    tag,
00418                    size);
00419 }
00420
00421 // ---
00422
00426 inline void set_device_memory_name(VkDevice device,
00427                                     VkDeviceMemory handle,
00428                                     name object) {
00429     set_object_name(device,
00430                     VK_OBJECT_TYPE_DEVICE_MEMORY,
00431                     VkObjectHandle(handle),
00432                     object);
00433 }
00434
00438 inline void set_device_memory_tag(VkDevice device,
00439                                    VkDeviceMemory handle,
00440                                    ui64 name,
00441                                    void_c_ptr tag,

```

```

00442                                     size_t size) {
00443     set_object_tag(device,
00444                   VK_OBJECT_TYPE_DEVICE_MEMORY,
00445                   VkObjectHandle(handle),
00446                   name,
00447                   tag,
00448                   size);
00449 }
00450
00451 // ---
00452
00456 inline void set_buffer_name(VkDevice device,
00457                             VkBuffer handle,
00458                             name object) {
00459     set_object_name(device,
00460                   VK_OBJECT_TYPE_BUFFER,
00461                   VkObjectHandle(handle),
00462                   object);
00463 }
00464
00468 inline void set_buffer_tag(VkDevice device,
00469                             VkBuffer handle,
00470                             ui64 name,
00471                             void_c_ptr tag,
00472                             size_t size) {
00473     set_object_tag(device,
00474                   VK_OBJECT_TYPE_BUFFER,
00475                   VkObjectHandle(handle),
00476                   name,
00477                   tag,
00478                   size);
00479 }
00480
00481 // ---
00482
00486 inline void set_image_name(VkDevice device,
00487                             VkImage handle,
00488                             name object) {
00489     set_object_name(device,
00490                   VK_OBJECT_TYPE_IMAGE,
00491                   VkObjectHandle(handle),
00492                   object);
00493 }
00494
00498 inline void set_image_tag(VkDevice device,
00499                             VkImage handle,
00500                             ui64 name,
00501                             void_c_ptr tag,
00502                             size_t size) {
00503     set_object_tag(device,
00504                   VK_OBJECT_TYPE_IMAGE,
00505                   VkObjectHandle(handle),
00506                   name,
00507                   tag,
00508                   size);
00509 }
00510
00511 // ---
00512
00516 inline void set_event_name(VkDevice device,
00517                             VkEvent handle,
00518                             name object) {
00519     set_object_name(device,
00520                   VK_OBJECT_TYPE_EVENT,
00521                   VkObjectHandle(handle),
00522                   object);
00523 }
00524
00528 inline void set_event_tag(VkDevice device,
00529                             VkEvent handle,
00530                             ui64 name,
00531                             void_c_ptr tag,
00532                             size_t size) {
00533     set_object_tag(device,
00534                   VK_OBJECT_TYPE_EVENT,
00535                   VkObjectHandle(handle),
00536                   name,
00537                   tag,
00538                   size);
00539 }
00540
00541 // ---
00542
00546 inline void set_query_pool_name(VkDevice device,
00547                                 VkQueryPool handle,
00548                                 name object) {
00549     set_object_name(device,

```

```

00550             VK_OBJECT_TYPE_QUERY_POOL,
00551             VkObjectHandle(handle),
00552             object);
00553 }
00554
00558 inline void set_query_pool_tag(VkDevice device,
00559                               VkQueryPool handle,
00560                               ui64 name,
00561                               void_c_ptr tag,
00562                               size_t size) {
00563     set_object_tag(device,
00564                   VK_OBJECT_TYPE_QUERY_POOL,
00565                   VkObjectHandle(handle),
00566                   name,
00567                   tag,
00568                   size);
00569 }
00570
00571 // ---
00572
00576 inline void set_buffer_view_name(VkDevice device,
00577                                  VkBufferView handle,
00578                                  name object) {
00579     set_object_name(device,
00580                   VK_OBJECT_TYPE_BUFFER_VIEW,
00581                   VkObjectHandle(handle),
00582                   object);
00583 }
00584
00588 inline void set_buffer_view_tag(VkDevice device,
00589                                  VkBufferView handle,
00590                                  ui64 name,
00591                                  void_c_ptr tag,
00592                                  size_t size) {
00593     set_object_tag(device,
00594                   VK_OBJECT_TYPE_BUFFER_VIEW,
00595                   VkObjectHandle(handle),
00596                   name,
00597                   tag,
00598                   size);
00599 }
00600
00601 // ---
00602
00606 inline void set_image_view_name(VkDevice device,
00607                                  VkImageView handle,
00608                                  name object) {
00609     set_object_name(device,
00610                   VK_OBJECT_TYPE_IMAGE_VIEW,
00611                   VkObjectHandle(handle),
00612                   object);
00613 }
00614
00618 inline void set_image_view_tag(VkDevice device,
00619                                  VkImageView handle,
00620                                  ui64 name,
00621                                  void_c_ptr tag,
00622                                  size_t size) {
00623     set_object_tag(device,
00624                   VK_OBJECT_TYPE_IMAGE_VIEW,
00625                   VkObjectHandle(handle),
00626                   name,
00627                   tag,
00628                   size);
00629 }
00630
00631 // ---
00632
00636 inline void set_shader_module_name(VkDevice device,
00637                                     VkShaderModule handle,
00638                                     name object) {
00639     set_object_name(device,
00640                   VK_OBJECT_TYPE_SHADER_MODULE,
00641                   VkObjectHandle(handle),
00642                   object);
00643 }
00644
00648 inline void set_shader_module_tag(VkDevice device,
00649                                     VkShaderModule handle,
00650                                     ui64 name,
00651                                     void_c_ptr tag,
00652                                     size_t size) {
00653     set_object_tag(device,
00654                   VK_OBJECT_TYPE_SHADER_MODULE,
00655                   VkObjectHandle(handle),
00656                   name,
00657                   tag,

```

```

00658             size);
00659 }
00660
00661 // ---
00662
00666 inline void set_pipeline_cache_name(VkDevice device,
00667                                     VkPipelineCache handle,
00668                                     name object) {
00669     set_object_name(device,
00670                     VK_OBJECT_TYPE_PIPELINE_CACHE,
00671                     VkObjectHandle(handle),
00672                     object);
00673 }
00674
00678 inline void set_pipeline_cache_tag(VkDevice device,
00679                                     VkPipelineCache handle,
00680                                     ui64 name,
00681                                     void_c_ptr tag,
00682                                     size_t size) {
00683     set_object_tag(device,
00684                    VK_OBJECT_TYPE_PIPELINE_CACHE,
00685                    VkObjectHandle(handle),
00686                    name,
00687                    tag,
00688                    size);
00689 }
00690
00691 // ---
00692
00696 inline void set_pipeline_layout_name(VkDevice device,
00697                                       VkPipelineLayout handle,
00698                                       name object) {
00699     set_object_name(device,
00700                     VK_OBJECT_TYPE_PIPELINE_LAYOUT,
00701                     VkObjectHandle(handle),
00702                     object);
00703 }
00704
00708 inline void set_pipeline_layout_tag(VkDevice device,
00709                                     VkPipelineLayout handle,
00710                                     ui64 name,
00711                                     void_c_ptr tag,
00712                                     size_t size) {
00713     set_object_tag(device,
00714                    VK_OBJECT_TYPE_PIPELINE_LAYOUT,
00715                    VkObjectHandle(handle),
00716                    name,
00717                    tag,
00718                    size);
00719 }
00720
00721 // ---
00722
00726 inline void set_render_pass_name(VkDevice device,
00727                                   VkRenderPass handle,
00728                                   name object) {
00729     set_object_name(device,
00730                     VK_OBJECT_TYPE_RENDER_PASS,
00731                     VkObjectHandle(handle),
00732                     object);
00733 }
00734
00738 inline void set_render_pass_tag(VkDevice device,
00739                                  VkRenderPass handle,
00740                                  ui64 name,
00741                                  void_c_ptr tag,
00742                                  size_t size) {
00743     set_object_tag(device,
00744                    VK_OBJECT_TYPE_RENDER_PASS,
00745                    VkObjectHandle(handle),
00746                    name,
00747                    tag,
00748                    size);
00749 }
00750
00751 // ---
00752
00756 inline void set_pipeline_name(VkDevice device,
00757                               VkPipeline handle,
00758                               name object) {
00759     set_object_name(device,
00760                     VK_OBJECT_TYPE_PIPELINE,
00761                     VkObjectHandle(handle),
00762                     object);
00763 }
00764
00768 inline void set_pipeline_tag(VkDevice device,

```

```

00769             VkPipeline handle,
00770             ui64 name,
00771             void_c_ptr tag,
00772             size_t size) {
00773     set_object_tag(device,
00774                   VK_OBJECT_TYPE_PIPELINE,
00775                   VkObjectHandle(handle),
00776                   name,
00777                   tag,
00778                   size);
00779 }
00780
00781 // ---
00782
00786 inline void set_descriptor_set_layout_name(VkDevice device,
00787                                           VkDescriptorSetLayout handle,
00788                                           name object) {
00789     set_object_name(device,
00790                   VK_OBJECT_TYPE_DESCRIPTOR_SET_LAYOUT,
00791                   VkObjectHandle(handle),
00792                   object);
00793 }
00794
00798 inline void set_descriptor_set_layout_tag(VkDevice device,
00799                                           VkDescriptorSetLayout handle,
00800                                           ui64 name,
00801                                           void_c_ptr tag,
00802                                           size_t size) {
00803     set_object_tag(device,
00804                   VK_OBJECT_TYPE_DESCRIPTOR_SET_LAYOUT,
00805                   VkObjectHandle(handle),
00806                   name,
00807                   tag,
00808                   size);
00809 }
00810
00811 // ---
00812
00816 inline void set_sampler_name(VkDevice device,
00817                               VkSampler handle,
00818                               name object) {
00819     set_object_name(device,
00820                   VK_OBJECT_TYPE_SAMPLER,
00821                   VkObjectHandle(handle),
00822                   object);
00823 }
00824
00828 inline void set_sampler_tag(VkDevice device,
00829                              VkSampler handle,
00830                              ui64 name,
00831                              void_c_ptr tag,
00832                              size_t size) {
00833     set_object_tag(device,
00834                   VK_OBJECT_TYPE_SAMPLER,
00835                   VkObjectHandle(handle),
00836                   name,
00837                   tag,
00838                   size);
00839 }
00840
00841 // ---
00842
00846 inline void set_descriptor_pool_name(VkDevice device,
00847                                       VkDescriptorPool handle,
00848                                       name object) {
00849     set_object_name(device,
00850                   VK_OBJECT_TYPE_DESCRIPTOR_POOL,
00851                   VkObjectHandle(handle),
00852                   object);
00853 }
00854
00858 inline void set_descriptor_pool_tag(VkDevice device,
00859                                      VkDescriptorPool handle,
00860                                      ui64 name,
00861                                      void_c_ptr tag,
00862                                      size_t size) {
00863     set_object_tag(device,
00864                   VK_OBJECT_TYPE_DESCRIPTOR_POOL,
00865                   VkObjectHandle(handle),
00866                   name,
00867                   tag,
00868                   size);
00869 }
00870
00871 // ---
00872
00876 inline void set_descriptor_set_name(VkDevice device,

```



```

00877             VkDescriptorSet handle,
00878             name object) {
00879     set_object_name(device,
00880                     VK_OBJECT_TYPE_DESCRIPTOR_SET,
00881                     VkObjectHandle(handle),
00882                     object);
00883 }
00884
00885 inline void set_descriptor_set_tag(VkDevice device,
00886                                   VkDescriptorSet handle,
00887                                   ui64 name,
00888                                   void_c_ptr tag,
00889                                   size_t size) {
00890     set_object_tag(device,
00891                   VK_OBJECT_TYPE_DESCRIPTOR_SET,
00892                   VkObjectHandle(handle),
00893                   name,
00894                   tag,
00895                   size);
00896 }
00897
00898 // ---
00899
00900 inline void set_framebuffer_name(VkDevice device,
00901                                  VkFramebuffer handle,
00902                                  name object) {
00903     set_object_name(device,
00904                     VK_OBJECT_TYPE_FRAMEBUFFER,
00905                     VkObjectHandle(handle),
00906                     object);
00907 }
00908
00909 inline void set_framebuffer_tag(VkDevice device,
00910                                 VkFramebuffer handle,
00911                                 ui64 name,
00912                                 void_c_ptr tag,
00913                                 size_t size) {
00914     set_object_tag(device,
00915                   VK_OBJECT_TYPE_FRAMEBUFFER,
00916                   VkObjectHandle(handle),
00917                   name,
00918                   tag,
00919                   size);
00920 }
00921
00922 // ---
00923
00924 inline void set_command_pool_name(VkDevice device,
00925                                   VkCommandPool handle,
00926                                   name object) {
00927     set_object_name(device,
00928                     VK_OBJECT_TYPE_COMMAND_POOL,
00929                     VkObjectHandle(handle),
00930                     object);
00931 }
00932
00933 inline void set_command_pool_tag(VkDevice device,
00934                                  VkCommandPool handle,
00935                                  ui64 name,
00936                                  void_c_ptr tag,
00937                                  size_t size) {
00938     set_object_tag(device,
00939                   VK_OBJECT_TYPE_COMMAND_POOL,
00940                   VkObjectHandle(handle),
00941                   name,
00942                   tag,
00943                   size);
00944 }
00945
00946 // ---
00947
00948 inline void set_sampler_ycbcr_conversion_name(VkDevice device,
00949                                                 VkSamplerYcbcrConversion handle,
00950                                                 name object) {
00951     set_object_name(device,
00952                     VK_OBJECT_TYPE_SAMPLER_YCBCR_CONVERSION,
00953                     VkObjectHandle(handle),
00954                     object);
00955 }
00956
00957 inline void set_sampler_ycbcr_conversion_tag(VkDevice device,
00958                                               VkSamplerYcbcrConversion handle,
00959                                               ui64 name,
00960                                               void_c_ptr tag,
00961                                               size_t size) {
00962     set_object_tag(device,
00963                   VK_OBJECT_TYPE_SAMPLER_YCBCR_CONVERSION,

```

```

00985         VkObjectHandle(handle),
00986         name,
00987         tag,
00988         size);
00989     }
00990
00991     // ---
00992
00996     inline void set_descriptor_update_template_name(VkDevice device,
00997                                                     VkDescriptorUpdateTemplate handle,
00998                                                     name object) {
00999         set_object_name(device,
01000                        VK_OBJECT_TYPE_DESCRIPTOR_UPDATE_TEMPLATE,
01001                        VkObjectHandle(handle),
01002                        object);
01003     }
01004
01008     inline void set_descriptor_update_template_tag(VkDevice device,
01009                                                    VkDescriptorUpdateTemplate handle,
01010                                                    ui64 name,
01011                                                    void_c_ptr tag,
01012                                                    size_t size) {
01013         set_object_tag(device,
01014                       VK_OBJECT_TYPE_DESCRIPTOR_UPDATE_TEMPLATE,
01015                       VkObjectHandle(handle),
01016                       name,
01017                       tag,
01018                       size);
01019     }
01020
01021     // ---
01022
01026     inline void set_surface_name(VkDevice device,
01027                                  VkSurfaceKHR handle,
01028                                  name object) {
01029         set_object_name(device,
01030                        VK_OBJECT_TYPE_SURFACE_KHR,
01031                        VkObjectHandle(handle),
01032                        object);
01033     }
01034
01038     inline void set_surface_tag(VkDevice device,
01039                                 VkSurfaceKHR handle,
01040                                 ui64 name,
01041                                 void_c_ptr tag,
01042                                 size_t size) {
01043         set_object_tag(device,
01044                       VK_OBJECT_TYPE_SURFACE_KHR,
01045                       VkObjectHandle(handle),
01046                       name,
01047                       tag,
01048                       size);
01049     }
01050
01051     // ---
01052
01056     inline void set_swapchain_name(VkDevice device,
01057                                    VkSwapchainKHR handle,
01058                                    name object) {
01059         set_object_name(device,
01060                        VK_OBJECT_TYPE_SWAPCHAIN_KHR,
01061                        VkObjectHandle(handle),
01062                        object);
01063     }
01064
01068     inline void set_swapchain_tag(VkDevice device,
01069                                   VkSwapchainKHR handle,
01070                                   ui64 name,
01071                                   void_c_ptr tag,
01072                                   size_t size) {
01073         set_object_tag(device,
01074                       VK_OBJECT_TYPE_SWAPCHAIN_KHR,
01075                       VkObjectHandle(handle),
01076                       name,
01077                       tag,
01078                       size);
01079     }
01080
01081     // ---
01082
01086     inline void set_display_name(VkDevice device,
01087                                  VkDisplayKHR handle,
01088                                  name object) {
01089         set_object_name(device,
01090                        VK_OBJECT_TYPE_DISPLAY_KHR,
01091                        VkObjectHandle(handle),
01092                        object);

```

```

01093 }
01094
01098 inline void set_display_tag(VkDevice device,
01099                             VkDisplayKHR handle,
01100                             ui64 name,
01101                             void_c_ptr tag,
01102                             size_t size) {
01103     set_object_tag(device,
01104                   VK_OBJECT_TYPE_DISPLAY_KHR,
01105                   VkObjectHandle(handle),
01106                   name,
01107                   tag,
01108                   size);
01109 }
01110
01111 // ---
01112
01116 inline void set_display_mode_name(VkDevice device,
01117                                   VkDisplayModeKHR handle,
01118                                   name object) {
01119     set_object_name(device,
01120                   VK_OBJECT_TYPE_DISPLAY_MODE_KHR,
01121                   VkObjectHandle(handle),
01122                   object);
01123 }
01124
01128 inline void set_display_mode_tag(VkDevice device,
01129                                  VkDisplayModeKHR handle,
01130                                  ui64 name,
01131                                  void_c_ptr tag,
01132                                  size_t size) {
01133     set_object_tag(device,
01134                   VK_OBJECT_TYPE_DISPLAY_MODE_KHR,
01135                   VkObjectHandle(handle),
01136                   name,
01137                   tag,
01138                   size);
01139 }
01140
01141 // ---
01142
01146 inline void set_debug_report_callback_name(VkDevice device,
01147                                             VkDebugReportCallbackEXT handle,
01148                                             name object) {
01149     set_object_name(device,
01150                   VK_OBJECT_TYPE_DEBUG_REPORT_CALLBACK_EXT,
01151                   VkObjectHandle(handle),
01152                   object);
01153 }
01154
01158 inline void set_debug_report_callback_tag(VkDevice device,
01159                                            VkDebugReportCallbackEXT handle,
01160                                            ui64 name,
01161                                            void_c_ptr tag,
01162                                            size_t size) {
01163     set_object_tag(device,
01164                   VK_OBJECT_TYPE_DEBUG_REPORT_CALLBACK_EXT,
01165                   VkObjectHandle(handle),
01166                   name,
01167                   tag,
01168                   size);
01169 }
01170
01171 // ---
01172
01176 inline void set_indirect_commands_layout_name(VkDevice device,
01177                                                VkIndirectCommandsLayoutNV handle,
01178                                                name object) {
01179     set_object_name(device,
01180                   VK_OBJECT_TYPE_INDIRECT_COMMANDS_LAYOUT_NV,
01181                   VkObjectHandle(handle),
01182                   object);
01183 }
01184
01188 inline void set_indirect_commands_layout_tag(VkDevice device,
01189                                               VkIndirectCommandsLayoutNV handle,
01190                                               ui64 name,
01191                                               void_c_ptr tag,
01192                                               size_t size) {
01193     set_object_tag(device,
01194                   VK_OBJECT_TYPE_INDIRECT_COMMANDS_LAYOUT_NV,
01195                   VkObjectHandle(handle),
01196                   name,
01197                   tag,
01198                   size);
01199 }
01200

```

```

01201 // ---
01202
01206 inline void set_debug_utils_messenger_name(VkDevice device,
01207                                             VkDebugUtilsMessengerEXT handle,
01208                                             name object) {
01209     set_object_name(device,
01210                    VK_OBJECT_TYPE_DEBUG_UTILS_MESSENGER_EXT,
01211                    VkObjectHandle(handle),
01212                    object);
01213 }
01214
01218 inline void set_debug_utils_messenger_tag(VkDevice device,
01219                                             VkDebugUtilsMessengerEXT handle,
01220                                             ui64 name,
01221                                             void_c_ptr tag,
01222                                             size_t size) {
01223     set_object_tag(device,
01224                   VK_OBJECT_TYPE_DEBUG_UTILS_MESSENGER_EXT,
01225                   VkObjectHandle(handle),
01226                   name,
01227                   tag,
01228                   size);
01229 }
01230
01231 // ---
01232
01236 inline void set_validation_cache_name(VkDevice device,
01237                                       VkValidationCacheEXT handle,
01238                                       name object) {
01239     set_object_name(device,
01240                    VK_OBJECT_TYPE_VALIDATION_CACHE_EXT,
01241                    VkObjectHandle(handle),
01242                    object);
01243 }
01244
01248 inline void set_validation_cache_tag(VkDevice device,
01249                                       VkValidationCacheEXT handle,
01250                                       ui64 name,
01251                                       void_c_ptr tag,
01252                                       size_t size) {
01253     set_object_tag(device,
01254                   VK_OBJECT_TYPE_VALIDATION_CACHE_EXT,
01255                   VkObjectHandle(handle),
01256                   name,
01257                   tag,
01258                   size);
01259 }
01260
01261 // ---
01262
01266 inline void set_acceleration_structure_nv_name(VkDevice device,
01267                                                 VkAccelerationStructureNV handle,
01268                                                 name object) {
01269     set_object_name(device,
01270                    VK_OBJECT_TYPE_ACCELERATION_STRUCTURE_NV,
01271                    VkObjectHandle(handle),
01272                    object);
01273 }
01274
01278 inline void set_acceleration_structure_nv_tag(VkDevice device,
01279                                               VkAccelerationStructureNV handle,
01280                                               ui64 name,
01281                                               void_c_ptr tag,
01282                                               size_t size) {
01283     set_object_tag(device,
01284                   VK_OBJECT_TYPE_ACCELERATION_STRUCTURE_NV,
01285                   VkObjectHandle(handle),
01286                   name,
01287                   tag,
01288                   size);
01289 }
01290
01291 // ---
01292
01296 inline void set_acceleration_structure_name(VkDevice device,
01297                                             VkAccelerationStructureKHR handle,
01298                                             name object) {
01299     set_object_name(device,
01300                    VK_OBJECT_TYPE_ACCELERATION_STRUCTURE_KHR,
01301                    VkObjectHandle(handle),
01302                    object);
01303 }
01304
01308 inline void set_acceleration_structure_tag(VkDevice device,
01309                                             VkAccelerationStructureKHR handle,
01310                                             ui64 name,
01311                                             void_c_ptr tag,

```

```

01312                                     size_t size) {
01313     set_object_tag(device,
01314                   VK_OBJECT_TYPE_ACCELERATION_STRUCTURE_KHR,
01315                   VkObjectHandle(handle),
01316                   name,
01317                   tag,
01318                   size);
01319 }
01320
01321 // ---
01322
01326 inline void set_performance_configuration_name(VkDevice device,
01327                                               VkPerformanceConfigurationINTEL handle,
01328                                               name object) {
01329     set_object_name(device,
01330                   VK_OBJECT_TYPE_PERFORMANCE_CONFIGURATION_INTEL,
01331                   VkObjectHandle(handle),
01332                   object);
01333 }
01334
01338 inline void set_performance_configuration_tag(VkDevice device,
01339                                               VkPerformanceConfigurationINTEL handle,
01340                                               ui64 name,
01341                                               void_c_ptr tag,
01342                                               size_t size) {
01343     set_object_tag(device,
01344                   VK_OBJECT_TYPE_PERFORMANCE_CONFIGURATION_INTEL,
01345                   VkObjectHandle(handle),
01346                   name,
01347                   tag,
01348                   size);
01349 }
01350
01351 // ---
01352
01356 inline void set_deferred_operation_name(VkDevice device,
01357                                         VkDeferredOperationKHR handle,
01358                                         name object) {
01359     set_object_name(device,
01360                   VK_OBJECT_TYPE_DEFERRED_OPERATION_KHR,
01361                   VkObjectHandle(handle),
01362                   object);
01363 }
01364
01368 inline void set_deferred_operation_tag(VkDevice device,
01369                                         VkDeferredOperationKHR handle,
01370                                         ui64 name,
01371                                         void_c_ptr tag,
01372                                         size_t size) {
01373     set_object_tag(device,
01374                   VK_OBJECT_TYPE_DEFERRED_OPERATION_KHR,
01375                   VkObjectHandle(handle),
01376                   name,
01377                   tag,
01378                   size);
01379 }
01380
01381 // ---
01382
01386 inline void set_private_data_slot_name(VkDevice device,
01387                                         VkPrivateDataSlotEXT handle,
01388                                         name object) {
01389     set_object_name(device,
01390                   VK_OBJECT_TYPE_PRIVATE_DATA_SLOT_EXT,
01391                   VkObjectHandle(handle),
01392                   object);
01393 }
01394
01398 inline void set_private_data_slot_tag(VkDevice device,
01399                                         VkPrivateDataSlotEXT handle,
01400                                         ui64 name,
01401                                         void_c_ptr tag,
01402                                         size_t size) {
01403     set_object_tag(device,
01404                   VK_OBJECT_TYPE_PRIVATE_DATA_SLOT_EXT,
01405                   VkObjectHandle(handle),
01406                   name,
01407                   tag,
01408                   size);
01409 }
01410
01411 } // namespace lava

```

## 5.33 liblava/base/device.hpp File Reference

Vulkan device.

```
#include "liblava/base/device_table.hpp"
#include "liblava/base/queue.hpp"
#include "liblava/core/data.hpp"
#include "liblava/core/id.hpp"
#include "liblava/fwd.hpp"
```

### Classes

- struct [lava::device](#)  
*Vulkan device.*
- struct [lava::device::create\\_param](#)  
*Device create parameters.*

### Typedefs

- using [lava::one\\_time\\_command\\_func](#) = std::function<void(VkCommandBuffer)>  
*One time command function.*

### Functions

- VkShaderModule [lava::create\\_shader\\_module](#) (device::ptr device, c\_data::ref data)  
*Create a shader module.*
- bool [lava::one\\_time\\_submit\\_pool](#) (device::ptr device, VkCommandPool pool, queue::ref queue, one\_time\_command\_func callback)  
*Submit one time command function with pool.*
- bool [lava::one\\_time\\_submit](#) (device::ptr device, queue::ref queue, one\_time\_command\_func callback)  
*Submit one time command function.*

### 5.33.1 Detailed Description

Vulkan device.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.33.2 Function Documentation

#### 5.33.2.1 create\_shader\_module()

```
VkShaderModule lava::create_shader_module (
    device::ptr device,
    c_data::ref data)
```

Create a shader module.

## Parameters

<i>device</i>	Vulkan device
<i>data</i>	Shader data

## Returns

VkShaderModule Shader module

### 5.33.2.2 one\_time\_submit()

```
bool lava::one_time_submit (
    device::ptr device,
    queue::ref queue,
    one_time_command_func callback) [inline]
```

Submit one time command function.

## Parameters

<i>device</i>	Vulkan device
<i>queue</i>	Target queue
<i>callback</i>	Function to be called

## Returns

Submit was successful or failed

### 5.33.2.3 one\_time\_submit\_pool()

```
bool lava::one_time_submit_pool (
    device::ptr device,
    VkCommandPool pool,
    queue::ref queue,
    one_time_command_func callback)
```

Submit one time command function with pool.

## Parameters

<i>device</i>	Vulkan device
<i>pool</i>	Command pool (VK_NULL_HANDLE: managed)
<i>queue</i>	Target queue
<i>callback</i>	Function to be called

## Returns

Submit was successful or failed

## 5.34 device.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/device_table.hpp"
00011 #include "liblava/base/queue.hpp"
00012 #include "liblava/core/data.hpp"
00013 #include "liblava/core/id.hpp"
00014 #include "liblava/fwd.hpp"
00015
00016 namespace lava {
00017
00021 struct device : device_table, entity {
00023     using ptr = device*;
00024
00026     using c_ptr = device const*;
00027
00029     using s_ptr = std::shared_ptr<device>;
00030
00032     using s_list = std::vector<s_ptr>;
00033
00035     using physical_device_c_ptr = physical_device const*;
00036
00040     struct create_param {
00042         using ref = create_param const&;
00043
00045         physical_device_c_ptr physical_device = nullptr;
00046
00048         VmaAllocatorCreateFlags vma_flags = 0;
00049
00051         names extensions;
00052
00054         VkPhysicalDeviceFeatures features{};
00055
00057         bool has_features_2 = false;
00058
00060         void const* next = nullptr;
00061
00063         queue_family_info::list queue_family_infos;
00064
00068         void add_swapchain_extension() {
00069             extensions.push_back("VK_KHR_swapchain");
00070         }
00071
00075         void add_portability_subset_extension() {
00076             extensions.push_back("VK_KHR_portability_subset");
00077         }
00078
00082         void set_default_queues() {
00083             lava::set_default_queues(queue_family_infos);
00084         }
00085
00089         void set_all_queues();
00090
00097         bool add_queue(VkQueueFlags flags,
00098             r32 priority = 1.f) {
00099             return add_queues(flags, 1, priority);
00100         }
00101
00109         bool add_queues(VkQueueFlags flags,
00110             ui32 count,
00111             r32 priority = 1.f);
00112
00118         bool add_dedicated_queues(r32 priority = 1.f);
00119
00124         verify_queues_result verify_queues() const;
00125     };
00126
00131     static s_ptr make() {
00132         return std::make_shared<device>();
00133     }
00134
00138     ~device() {
00139         destroy();
00140     }
00141
00147     bool create(create_param::ref param);
00148
00152     void destroy();
00153
00159     queue::ref get_graphics_queue(index index = 0) const {
00160         return get_graphics_queues().at(index);
00161     }

```



```

00162
00166     queue::ref graphics_queue(index index = 0) const {
00167         return get_graphics_queue(index);
00168     }
00169
00175     queue::ref get_compute_queue(index index = 0) const {
00176         return get_compute_queues().at(index);
00177     }
00178
00182     queue::ref compute_queue(index index = 0) const {
00183         return get_compute_queue(index);
00184     }
00185
00191     queue::ref get_transfer_queue(index index = 0) const {
00192         return get_transfer_queues().at(index);
00193     }
00194
00198     queue::ref transfer_queue(index index = 0) const {
00199         return get_transfer_queue(index);
00200     }
00201
00206     queue::list const& get_graphics_queues() const {
00207         return m_graphics_queue_list;
00208     }
00209
00213     queue::list const& graphics_queues() const {
00214         return get_graphics_queues();
00215     }
00216
00221     queue::list const& get_compute_queues() const {
00222         return m_compute_queue_list;
00223     }
00224
00228     queue::list const& compute_queues() const {
00229         return get_compute_queues();
00230     }
00231
00236     queue::list const& get_transfer_queues() const {
00237         return m_transfer_queue_list;
00238     }
00239
00243     queue::list const& transfer_queues() const {
00244         return get_transfer_queues();
00245     }
00246
00251     queue::list const& get_queues() const {
00252         return m_queue_list;
00253     }
00254
00258     queue::list const& queues() const {
00259         return get_queues();
00260     }
00261
00266     VkDevice get() const {
00267         return vk_device;
00268     }
00269
00274     VolkDeviceTable const& call() const {
00275         return table;
00276     }
00277
00282     bool wait_for_idle() const {
00283         return check(call().vkDeviceWaitIdle(vk_device));
00284     }
00285
00290     physical_device_c_ptr get_physical_device() const {
00291         return m_physical_device;
00292     }
00293
00298     VkPhysicalDevice get_vk_physical_device() const;
00299
00304     VkPhysicalDeviceFeatures const& get_features() const;
00305
00310     VkPhysicalDeviceProperties const& get_properties() const;
00311
00317     bool surface_supported(VkSurfaceKHR surface) const;
00318
00323     void set_allocator(allocator::s_ptr value) {
00324         m_mem_allocator = value;
00325     }
00326
00331     allocator::s_ptr get_allocator() {
00332         return m_mem_allocator;
00333     }
00334
00339     VmaAllocator alloc() const {
00340         return m_mem_allocator != nullptr

```

```

00341             ? m_mem_allocator->get()
00342             : nullptr;
00343     }
00344
00345 private:
00346     physical_device_c_ptr m_physical_device = nullptr;
00347     queue::list m_graphics_queue_list;
00348     queue::list m_compute_queue_list;
00349     queue::list m_transfer_queue_list;
00350     queue::list m_queue_list;
00351
00352     VkPhysicalDeviceFeatures m_features{};
00353     allocator::s_ptr m_mem_allocator;
00354 };
00355
00356 VkShaderModule create_shader_module(device::ptr device,
00357                                     c_data::ref data);
00358
00359 using one_time_command_func = std::function<void(VkCommandBuffer)>;
00360
00361 bool one_time_submit_pool(device::ptr device,
00362                           VkCommandPool pool,
00363                           queue::ref queue,
00364                           one_time_command_func callback);
00365
00366 inline bool one_time_submit(device::ptr device,
00367                             queue::ref queue,
00368                             one_time_command_func callback) {
00369     return one_time_submit_pool(device,
00370                                 VK_NULL_HANDLE,
00371                                 queue,
00372                                 callback);
00373 }
00374
00375 } // namespace lava

```

## 5.35 liblava/base/device\_table.hpp File Reference

Device functions.

```
#include "liblava/base/memory.hpp"
```

### Classes

- struct [lava::device\\_table](#)

*Device functions.*

### 5.35.1 Detailed Description

Device functions.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.36 device\_table.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/memory.hpp"
00011
00012 namespace lava {
00013
00017 struct device_table {
00021     void load_table() {
00022         volkLoadDeviceTable(&table, vk_device);
00023     }
00024
00028     vk_result vkCreateImageView(const VkImageViewCreateInfo* pCreateInfo,
00029                               const VkAllocationCallbacks* pAllocator,
00030                               VkImageView* pView) {
00031         auto result = table.vkCreateImageView(vk_device,
00032                                             pCreateInfo,
00033                                             pAllocator,
00034                                             pView);
00035         return {check(result), result};
00036     }
00037
00041     vk_result vkCreateImageView(const VkImageViewCreateInfo* pCreateInfo,
00042                               VkImageView* pView) {
00043         return vkCreateImageView(pCreateInfo,
00044                                 memory::instance().alloc(),
00045                                 pView);
00046     }
00047
00051     vk_result vkCreateSampler(const VkSamplerCreateInfo* pCreateInfo,
00052                             const VkAllocationCallbacks* pAllocator,
00053                             VkSampler* pSampler) {
00054         auto result = table.vkCreateSampler(vk_device,
00055                                             pCreateInfo,
00056                                             pAllocator,
00057                                             pSampler);
00058         return {check(result), result};
00059     }
00060
00064     vk_result vkCreateSampler(const VkSamplerCreateInfo* pCreateInfo,
00065                             VkSampler* pSampler) {
00066         return vkCreateSampler(pCreateInfo,
00067                                 memory::instance().alloc(),
00068                                 pSampler);
00069     }
00070
00074     vk_result vkCreateShaderModule(const VkShaderModuleCreateInfo* pCreateInfo,
00075                                   const VkAllocationCallbacks* pAllocator,
00076                                   VkShaderModule* pShaderModule) {
00077         auto result = table.vkCreateShaderModule(vk_device,
00078                                                 pCreateInfo,
00079                                                 pAllocator,
00080                                                 pShaderModule);
00081         return {check(result), result};
00082     }
00083
00087     vk_result vkCreateShaderModule(const VkShaderModuleCreateInfo* pCreateInfo,
00088                                   VkShaderModule* pShaderModule) {
00089         return vkCreateShaderModule(pCreateInfo,
00090                                     memory::instance().alloc(),
00091                                     pShaderModule);
00092     }
00093
00097     vk_result vkCreateFence(const VkFenceCreateInfo* pCreateInfo,
00098                           const VkAllocationCallbacks* pAllocator,
00099                           VkFence* pFence) {
00100         auto result = table.vkCreateFence(vk_device,
00101                                           pCreateInfo,
00102                                           pAllocator,
00103                                           pFence);
00104         return {check(result), result};
00105     }
00106
00110     vk_result vkCreateFence(const VkFenceCreateInfo* pCreateInfo,
00111                           VkFence* pFence) {
00112         return vkCreateFence(pCreateInfo,
00113                             memory::instance().alloc(),
00114                             pFence);
00115     }
00116
00120     vk_result vkCreateSemaphore(const VkSemaphoreCreateInfo* pCreateInfo,
00121                                const VkAllocationCallbacks* pAllocator,

```

```

00122         VkSemaphore* pSemaphore) {
00123     auto result = table.vkCreateSemaphore(vk_device,
00124                                           pCreateInfo,
00125                                           pAllocator,
00126                                           pSemaphore);
00127     return {check(result), result};
00128 }
00129
00130 vk_result vkCreateSemaphore(const VkSemaphoreCreateInfo* pCreateInfo,
00131                             VkSemaphore* pSemaphore) {
00132     return vkCreateSemaphore(pCreateInfo,
00133                             memory::instance().alloc(),
00134                             pSemaphore);
00135 }
00136
00137 vk_result vkWaitForFences(uint32_t fenceCount,
00138                           const VkFence* pFences,
00139                           VkBool32 waitAll,
00140                           uint64_t timeout) {
00141     auto result = table.vkWaitForFences(vk_device,
00142                                         fenceCount,
00143                                         pFences,
00144                                         waitAll,
00145                                         timeout);
00146     if ((result == VK_TIMEOUT) && (timeout != UINT64_MAX))
00147         return {false, result};
00148     return {check(result), result};
00149 }
00150
00151 vk_result vkResetFences(uint32_t fenceCount,
00152                         const VkFence* pFences) {
00153     auto result = table.vkResetFences(vk_device,
00154                                       fenceCount,
00155                                       pFences);
00156     return {check(result), result};
00157 }
00158
00159 vk_result vkQueueSubmit(VkQueue queue,
00160                        uint32_t submitCount,
00161                        const VkSubmitInfo* pSubmits,
00162                        VkFence fence) {
00163     auto result = table.vkQueueSubmit(queue,
00164                                       submitCount,
00165                                       pSubmits,
00166                                       fence);
00167     return {check(result), result};
00168 }
00169
00170 vk_result vkAcquireNextImageKHR(VkSwapchainKHR swapchain,
00171                                uint64_t timeout,
00172                                VkSemaphore semaphore,
00173                                VkFence fence,
00174                                uint32_t* pImageIndex) {
00175     auto result = table.vkAcquireNextImageKHR(vk_device,
00176                                               swapchain,
00177                                               timeout,
00178                                               semaphore,
00179                                               fence,
00180                                               pImageIndex);
00181     return {check(result), result};
00182 }
00183
00184 vk_result vkQueuePresentKHR(VkQueue queue,
00185                            const VkPresentInfoKHR* pPresentInfo) {
00186     auto result = table.vkQueuePresentKHR(queue,
00187                                           pPresentInfo);
00188     return {check(result), result};
00189 }
00190
00191 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00192                               const VkAllocationCallbacks* pAllocator,
00193                               VkSwapchainKHR* pSwapchain) {
00194     auto result = table.vkCreateSwapchainKHR(vk_device,
00195                                              pCreateInfo,
00196                                              pAllocator,
00197                                              pSwapchain);
00198     return {check(result), result};
00199 }
00200
00201 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00202                               VkSwapchainKHR* pSwapchain) {
00203     return vkCreateSwapchainKHR(pCreateInfo,
00204                                 memory::instance().alloc(),
00205                                 pSwapchain);
00206 }
00207
00208 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00209                               VkSwapchainKHR* pSwapchain) {
00210     return vkCreateSwapchainKHR(pCreateInfo,
00211                                 memory::instance().alloc(),
00212                                 pSwapchain);
00213 }
00214
00215 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00216                               VkSwapchainKHR* pSwapchain) {
00217     return vkCreateSwapchainKHR(pCreateInfo,
00218                                 memory::instance().alloc(),
00219                                 pSwapchain);
00220 }
00221
00222 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00223                               VkSwapchainKHR* pSwapchain) {
00224     return vkCreateSwapchainKHR(pCreateInfo,
00225                                 memory::instance().alloc(),
00226                                 pSwapchain);
00227 }
00228
00229 vk_result vkCreateSwapchainKHR(const VkSwapchainCreateInfoKHR* pCreateInfo,
00230                               VkSwapchainKHR* pSwapchain) {
00231     return vkCreateSwapchainKHR(pCreateInfo,
00232                                 memory::instance().alloc(),
00233                                 pSwapchain);
00234 }

```

```

00236 void vkDestroySwapchainKHR(VkSwapchainKHR swapchain,
00237                             const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00238     table.vkDestroySwapchainKHR(vk_device,
00239                                 swapchain,
00240                                 pAllocator);
00241 }
00242
00246 vk_result vkGetSwapchainImagesKHR(VkSwapchainKHR swapchain,
00247                                   uint32_t* pSwapchainImageCount, VkImage* pSwapchainImages) {
00248     auto result = table.vkGetSwapchainImagesKHR(vk_device,
00249                                                 swapchain,
00250                                                 pSwapchainImageCount,
00251                                                 pSwapchainImages);
00252     return {check(result), result};
00253 }
00254
00258 vk_result vkCreateCommandPool(const VkCommandPoolCreateInfo* pCreateInfo,
00259                               const VkAllocationCallbacks* pAllocator,
00260                               VkCommandPool* pCommandPool) {
00261     auto result = table.vkCreateCommandPool(vk_device,
00262                                             pCreateInfo,
00263                                             pAllocator,
00264                                             pCommandPool);
00265     return {check(result), result};
00266 }
00267
00271 vk_result vkCreateCommandPool(const VkCommandPoolCreateInfo* pCreateInfo,
00272                               VkCommandPool* pCommandPool) {
00273     return vkCreateCommandPool(pCreateInfo,
00274                                memory::instance().alloc(),
00275                                pCommandPool);
00276 }
00277
00281 vk_result vkCreateCommandPool(index queue_family,
00282                               VkCommandPool* pCommandPool) {
00283     VkCommandPoolCreateInfo create_info{
00284         .sType = VK_STRUCTURE_TYPE_COMMAND_POOL_CREATE_INFO,
00285         .queueFamilyIndex = queue_family,
00286     };
00287     return vkCreateCommandPool(&create_info,
00288                                pCommandPool);
00289 }
00290
00294 vk_result vkAllocateCommandBuffers(const VkCommandBufferAllocateInfo* pAllocateInfo,
00295                                   VkCommandBuffer* pCommandBuffers) {
00296     auto result = table.vkAllocateCommandBuffers(vk_device,
00297                                                 pAllocateInfo,
00298                                                 pCommandBuffers);
00299     return {check(result), result};
00300 }
00301
00305 vk_result vkAllocateCommandBuffers(VkCommandPool commandPool,
00306                                   uint32_t commandBufferCount,
00307                                   VkCommandBuffer* pCommandBuffers,
00308                                   VkCommandBufferLevel level = VK_COMMAND_BUFFER_LEVEL_PRIMARY) {
00309     VkCommandBufferAllocateInfo alloc_info{
00310         .sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_ALLOCATE_INFO,
00311         .commandPool = commandPool,
00312         .level = level,
00313         .commandBufferCount = commandBufferCount,
00314     };
00315     return vkAllocateCommandBuffers(&alloc_info,
00316                                    pCommandBuffers);
00317 }
00318
00322 void vkDestroyImageView(VkImageView imageView,
00323                         const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00324     table.vkDestroyImageView(vk_device,
00325                             imageView,
00326                             pAllocator);
00327 }
00328
00332 void vkDestroyFence(VkFence fence,
00333                     const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00334     table.vkDestroyFence(vk_device,
00335                          fence,
00336                          pAllocator);
00337 }
00338
00342 void vkDestroySemaphore(VkSemaphore semaphore,
00343                         const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00344     table.vkDestroySemaphore(vk_device,
00345                             semaphore,
00346                             pAllocator);
00347 }
00348
00352 void vkFreeCommandBuffers(VkCommandPool commandPool,

```

```

00353             uint32_t commandBufferCount,
00354             const VkCommandBuffer* pCommandBuffers) {
00355         table.vkFreeCommandBuffers(vk_device,
00356             commandPool,
00357             commandBufferCount,
00358             pCommandBuffers);
00359     }
00360
00361 void vkDestroyCommandPool(VkCommandPool commandPool,
00362     const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00363     table.vkDestroyCommandPool(vk_device,
00364         commandPool,
00365         pAllocator);
00366 }
00367
00368 void vkDestroySampler(VkSampler sampler,
00369     const VkAllocationCallbacks* pAllocator = memory::instance().alloc()) {
00370     table.vkDestroySampler(vk_device,
00371         sampler,
00372         pAllocator);
00373 }
00374
00375 void vkUpdateDescriptorSets(uint32_t descriptorWriteCount,
00376     const VkWriteDescriptorSet* pDescriptorWrites,
00377     uint32_t descriptorCopyCount = 0,
00378     const VkCopyDescriptorSet* pDescriptorCopies = nullptr) {
00379     table.vkUpdateDescriptorSets(vk_device,
00380         descriptorWriteCount,
00381         pDescriptorWrites,
00382         descriptorCopyCount,
00383         pDescriptorCopies);
00384 }
00385
00386 template <std::size_t SIZE>
00387 void vkUpdateDescriptorSets(std::array<VkWriteDescriptorSet, SIZE> const& descriptor_writes) {
00388     vkUpdateDescriptorSets(to_i32(descriptor_writes.size()),
00389         descriptor_writes.data());
00390 }
00391
00392 template <std::size_t SIZE>
00393 void vkUpdateDescriptorSets(std::array<VkCopyDescriptorSet, SIZE> const& descriptor_copies) {
00394     vkUpdateDescriptorSets(0,
00395         nullptr,
00396         to_i32(descriptor_copies.size()),
00397         descriptor_copies.data());
00398 }
00399
00400 template <std::size_t WRITE_SIZE, std::size_t COPY_SIZE>
00401 void vkUpdateDescriptorSets(std::array<VkWriteDescriptorSet, WRITE_SIZE> const& descriptor_writes,
00402     std::array<VkCopyDescriptorSet, COPY_SIZE> const& descriptor_copies) {
00403     vkUpdateDescriptorSets(to_i32(descriptor_writes.size()),
00404         descriptor_writes.data(),
00405         to_i32(descriptor_copies.size()),
00406         descriptor_copies.data());
00407 }
00408
00409 void vkUpdateDescriptorSets(std::initializer_list<VkWriteDescriptorSet> descriptor_writes) {
00410     vkUpdateDescriptorSets(to_i32(descriptor_writes.size()),
00411         descriptor_writes.begin());
00412 }
00413
00414 void vkUpdateDescriptorSets(std::initializer_list<VkCopyDescriptorSet> descriptor_copies) {
00415     vkUpdateDescriptorSets(0,
00416         nullptr,
00417         to_i32(descriptor_copies.size()),
00418         descriptor_copies.begin());
00419 }
00420
00421 void vkUpdateDescriptorSets(std::initializer_list<VkWriteDescriptorSet> descriptor_writes,
00422     std::initializer_list<VkCopyDescriptorSet> descriptor_copies) {
00423     vkUpdateDescriptorSets(to_i32(descriptor_writes.size()),
00424         descriptor_writes.begin(),
00425         to_i32(descriptor_copies.size()),
00426         descriptor_copies.begin());
00427 }
00428
00429 VkDevice vk_device = nullptr;
00430 VolkDeviceTable table = {};
00431 };
00432
00433 } // namespace lava

```

## 5.37 liblava/base/instance.hpp File Reference

Vulkan instance.

```
#include "liblava/base/physical_device.hpp"
```

### Classes

- struct [lava::instance\\_info](#)  
*Vulkan instance information.*
- struct [lava::instance](#)  
*Vulkan instance.*
- struct [lava::instance::create\\_param](#)  
*Instance create parameters.*
- struct [lava::instance::debug\\_config](#)  
*Debug configuration.*

### Functions

- bool [lava::check](#) ([instance::create\\_param::ref](#) param)  
*Check instance create parameters.*
- [sem\\_version](#) [lava::get\\_instance\\_version](#) ()  
*Get the instance version.*
- [VkLayerPropertiesList](#) [lava::enumerate\\_layer\\_properties](#) ()  
*Enumerate enabled layer properties.*
- [VkExtensionPropertiesList](#) [lava::enumerate\\_extension\\_properties](#) (name layer\_name=nullptr)  
*Enumerate enabled extension properties.*

### 5.37.1 Detailed Description

Vulkan instance.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.37.2 Function Documentation

#### 5.37.2.1 [check\(\)](#)

```
bool lava::check (
    instance::create\_param::ref param)
```

Check instance create parameters.

**Parameters**

<i>param</i>	Create parameters
--------------	-------------------

**Returns**

Check was successful or failed

**5.37.2.2 enumerate\_extension\_properties()**

```
VkExtensionPropertiesList lava::enumerate_extension_properties (  
    name layer_name = nullptr)
```

Enumerate enabled extension properties.

**Parameters**

<i>layer_name</i>	Name of layer
-------------------	---------------

**Returns**

VkExtensionPropertiesList List of extension properties

**5.37.2.3 enumerate\_layer\_properties()**

```
VkLayerPropertiesList lava::enumerate_layer_properties ()
```

Enumerate enabled layer properties.

**Returns**

VkLayerPropertiesList List of layer properties

**5.37.2.4 get\_instance\_version()**

```
sem_version lava::get_instance_version ()
```

Get the instance version.

**Returns**

sem\_version Version



## 5.38 instance.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/physical_device.hpp"
00011
00012 namespace lava {
00013
00017 struct instance_info {
00019     using ref = instance_info const&;
00020
00022     name app_name = _lava_;
00023
00025     name engine_name = _liblava_;
00026
00028     sem_version app_version;
00029
00031     sem_version engine_version;
00032
00034     api_version req_api_version = api_version::v1_0;
00035 };
00036
00040 struct instance : no_copy_no_move {
00044     struct create_param {
00046         using ref = create_param const&;
00047
00049         names layers{};
00050
00052         names extensions{};
00053     };
00054
00058     struct debug_config {
00060         using ref = debug_config const&;
00061
00063         bool validation = false;
00064
00066         bool render_doc = false;
00067
00069         bool verbose = false;
00070
00072         bool utils = false;
00073     };
00074
00079     static instance& singleton() {
00080         static instance instance;
00081         return instance;
00082     }
00083
00091     bool create(create_param& param,
00092               debug_config::ref debug,
00093               instance_info::ref info);
00094
00098     void destroy();
00099
00104     physical_device::s_list const& get_physical_devices() const {
00105         return m_physical_devices;
00106     }
00107
00112     physical_device::ref get_first_physical_device() const {
00113         return *m_physical_devices.front().get();
00114     }
00115
00120     VkInstance get() const {
00121         return m_vk_instance;
00122     }
00123
00128     debug_config::ref get_debug_config() const {
00129         return m_debug;
00130     }
00131
00136     instance_info::ref get_info() const {
00137         return m_info;
00138     }
00139
00140 private:
00144     explicit instance() = default;
00145
00149     ~instance();
00150
00156     bool check_debug(create_param& param) const;
00157
00162     bool enumerate_physical_devices();
00163

```

```

00168     bool create_debug_messenger();
00169
00173     void destroy_debug_messenger();
00174
00176     VkInstance m_vk_instance = nullptr;
00177
00179     physical_device::s_list m_physical_devices;
00180
00182     debug_config m_debug;
00183
00185     instance_info m_info;
00186
00188     VkDebugUtilsMessengerEXT m_debug_messenger = VK_NULL_HANDLE;
00189 };
00190
00196 bool check(instance::create_param::ref param);
00197
00202 sem_version get_instance_version();
00203
00208 VkLayerPropertiesList enumerate_layer_properties();
00209
00215 VkExtensionPropertiesList enumerate_extension_properties(name layer_name = nullptr);
00216
00217 } // namespace lava

```

## 5.39 liblava/base/memory.hpp File Reference

Vulkan allocator.

```

#include "liblava/base/base.hpp"
#include "liblava/fwd.hpp"
#include "vk_mem_alloc.h"
#include <memory>

```

### Classes

- struct [lava::allocator](#)  
*Vulkan allocator.*
- struct [lava::memory](#)  
*Vulkan memory.*

### Functions

- [allocator::s\\_ptr lava::create\\_allocator](#) ([allocator::device\\_c\\_ptr device](#), VmaAllocatorCreateFlags flags=0)  
*Create a allocator.*
- [index lava::find\\_memory\\_type\\_with\\_properties](#) (VkPhysicalDeviceMemoryProperties properties, [ui32 type\\_bits](#), VkMemoryPropertyFlags required\_properties)  
*Find the memory type with properties.*
- [index lava::find\\_memory\\_type](#) (VkPhysicalDevice gpu, VkMemoryPropertyFlags properties, [ui32 type\\_bits](#))  
*Find the memory type.*

### 5.39.1 Detailed Description

Vulkan allocator.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.39.2 Function Documentation

### 5.39.2.1 create\_allocator()

```
allocator::s_ptr lava::create_allocator (
    allocator::device_c_ptr device,
    VmaAllocatorCreateFlags flags = 0) [inline]
```

Create a allocator.

#### Parameters

<i>device</i>	Vulkan device
<i>flags</i>	VMA allocator create flags

#### Returns

allocator::s\_ptr Allocator

### 5.39.2.2 find\_memory\_type()

```
index lava::find_memory_type (
    VkPhysicalDevice gpu,
    VkMemoryPropertyFlags properties,
    ui32 type_bits)
```

Find the memory type.

#### Parameters

<i>gpu</i>	Physical device
<i>properties</i>	Memory properties flags
<i>type_bits</i>	Type bits

#### Returns

type Result type

### 5.39.2.3 find\_memory\_type\_with\_properties()

```
index lava::find_memory_type_with_properties (
    VkPhysicalDeviceMemoryProperties properties,
    ui32 type_bits,
    VkMemoryPropertyFlags required_properties)
```

Find the memory type with properties.

## Parameters

<i>properties</i>	Physical device memory properties
<i>type_bits</i>	Type bits
<i>required_properties</i>	Memory property flags

## Returns

type Result type

## 5.40 memory.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 // clang-format off
00011
00012 #include "liblava/base/base.hpp"
00013 #include "liblava/fwd.hpp"
00014 #include "vk_mem_alloc.h"
00015 #include <memory>
00016
00017 // clang-format on
00018
00019 namespace lava {
00020
00024 struct allocator {
00026     using s_ptr = std::shared_ptr<allocator>;
00027
00029     using device_c_ptr = device const*;
00030
00035     static s_ptr make() {
00036         return std::make_shared<allocator>();
00037     }
00038
00042     allocator() = default;
00043
00048     explicit allocator(VmaAllocator allocator)
00049     : m_allocator(allocator) {}
00050
00057     bool create(device_c_ptr device,
00058                VmaAllocatorCreateFlags flags = 0);
00059
00063     void destroy();
00064
00069     bool valid() const {
00070         return m_allocator != nullptr;
00071     }
00072
00077     VmaAllocator get() const {
00078         return m_allocator;
00079     }
00080
00081 private:
00083     VmaAllocator m_allocator = nullptr;
00084 };
00085
00092 inline allocator::s_ptr create_allocator(allocator::device_c_ptr device,
00093                                         VmaAllocatorCreateFlags flags = 0) {
00094     auto result = allocator::make();
00095     if (!result->create(device, flags))
00096         return nullptr;
00097
00098     return result;
00099 }
00100
00104 struct memory : no_copy_no_move {
00108     memory();
00109
00114     static memory& instance() {
00115         static memory memory;
00116         return memory;
00117     }
00118

```

```

00123     VkAllocationCallbacks* alloc() {
00124         if (m_use_custom_cpu_callbacks)
00125             return &m_vk_callbacks;
00126
00127         return nullptr;
00128     }
00129
00134     void set_callbacks(VkAllocationCallbacks const& callbacks) {
00135         m_vk_callbacks = callbacks;
00136     }
00137
00142     void set_use_custom_cpu_callbacks(bool value) {
00143         m_use_custom_cpu_callbacks = value;
00144     }
00145
00146 private:
00148     bool m_use_custom_cpu_callbacks = true;
00149
00151     VkAllocationCallbacks m_vk_callbacks = {};
00152 };
00153
00161 index find_memory_type_with_properties(VkPhysicalDeviceMemoryProperties properties,
00162                                         ui32 type_bits,
00163                                         VkMemoryPropertyFlags required_properties);
00164
00172 index find_memory_type(VkPhysicalDevice gpu,
00173                         VkMemoryPropertyFlags properties,
00174                         ui32 type_bits);
00175
00176 } // namespace lava

```

## 5.41 liblava/base/physical\_device.hpp File Reference

Vulkan physical device.

```
#include "liblava/base/device.hpp"
```

### Classes

- struct [lava::physical\\_device](#)  
*Vulkan physical device.*

### 5.41.1 Detailed Description

Vulkan physical device.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.42 physical\_device.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct physical_device : entity {
00019     using s_ptr = std::shared_ptr<physical_device>;
00020
00022     using s_list = std::vector<s_ptr>;
00023
00025     using ref = physical_device const&;
00026
00032     static s_ptr make(VkPhysicalDevice vk_physical_device) {
00033         return std::make_shared<physical_device>(vk_physical_device);
00034     }
00035
00039     physical_device() = default;
00040
00045     physical_device(VkPhysicalDevice vk_physical_device);
00046
00051     void initialize(VkPhysicalDevice vk_physical_device);
00052
00058     bool supported(string_ref extension) const;
00059
00066     bool get_queue_family(index& index, VkQueueFlags flags) const;
00067
00072     device::create_param create_default_device_param() const;
00073
00078     VkPhysicalDeviceProperties const& get_properties() const {
00079         return m_properties;
00080     }
00081
00086     VkPhysicalDeviceFeatures const& get_features() const {
00087         return m_features;
00088     }
00089
00094     VkPhysicalDeviceMemoryProperties const& get_memory_properties() const {
00095         return m_memory_properties;
00096     }
00097
00102     VkQueueFamilyPropertiesList const& get_queue_family_properties() const {
00103         return m_queue_family_properties;
00104     }
00105
00110     VkExtensionPropertiesList const& get_extension_properties() const {
00111         return m_extension_properties;
00112     }
00113
00118     VkPhysicalDevice get() const {
00119         return m_vk_physical_device;
00120     }
00121
00126     name get_device_name() const;
00127
00132     string get_device_type_string() const;
00133
00138     sem_version get_driver_version() const;
00139
00144     bool swapchain_supported() const;
00145
00152     bool surface_supported(index queue_family,
00153                           VkSurfaceKHR surface) const;
00154
00155 private:
00157     VkPhysicalDevice m_vk_physical_device = nullptr;
00158
00160     VkPhysicalDeviceProperties m_properties = {};
00161
00163     VkPhysicalDeviceFeatures m_features = {};
00164
00166     VkPhysicalDeviceMemoryProperties m_memory_properties = {};
00167
00169     VkQueueFamilyPropertiesList m_queue_family_properties;
00170
00172     VkExtensionPropertiesList m_extension_properties;
00173 };
00174
00175 } // namespace lava

```

## 5.43 liblava/base/platform.hpp File Reference

Stage platform.

```
#include "liblava/base/device.hpp"
```

### Classes

- struct [lava::platform](#)  
*Stage platform.*

### 5.43.1 Detailed Description

Stage platform.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.44 platform.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct platform {
00019     using ptr = platform*;
00020
00026     device::s_ptr create(index physical_device = 0);
00027
00033     device::s_ptr create(device::create_param::ref param);
00034
00040     device::ptr create_device(index physical_device = 0);
00041
00046     device::s_list const& get_devices() const {
00047         return m_devices;
00048     }
00049
00053     void wait_idle();
00054
00060     bool remove(id::ref device_id);
00061
00065     void clear();
00066
00068     using create_param_func = std::function<void(device::create_param&)>;
00069
00071     create_param_func on_create_param;
00072
00073 private:
00075     device::s_list m_devices;
00076 };
00077
00078 } // namespace lava
```

## 5.45 liblava/base/queue.hpp File Reference

Device queue.

```
#include "liblava/base/base.hpp"
#include <deque>
```

### Classes

- struct [lava::queue](#)  
*Device queue.*
- struct [lava::queue\\_info](#)  
*Queue information.*
- struct [lava::queue\\_family\\_info](#)  
*Queue family information.*

### Enumerations

- enum class [lava::verify\\_queues\\_result](#) : index {  
  **ok** = 0 , **empty\_list** , **no\_properties** , **duplicate\_family\_index** ,  
  **no\_family\_index** , **no\_queues** , **too\_many\_queues** , **no\_compatible\_flags** }  
*Result of queue verifications.*

### Functions

- void [lava::set\\_default\\_queues](#) (queue\_family\_info::list &list)  
*Set the default queues.*
- void [lava::set\\_all\\_queues](#) (queue\_family\_info::list &list, VkQueueFamilyPropertiesList const &properties)  
*Set all queues.*
- bool [lava::add\\_queues](#) (queue\_family\_info::list &list, VkQueueFamilyPropertiesList const &properties, VkQueueFlags flags, ui32 count, r32 priority=1.f)  
*Add queues.*
- bool [lava::add\\_dedicated\\_queues](#) (queue\_family\_info::list &list, VkQueueFamilyPropertiesList const &properties, r32 priority=1.f)  
*Add dedicated queues.*
- [verify\\_queues\\_result](#) [lava::verify\\_queues](#) (queue\_family\_info::list const &list, VkQueueFamilyPropertiesList const &properties)  
*Verify queues.*

### Variables

- constexpr VkQueueFlags const [lava::default\\_queue\\_flags](#)  
*Default queue flags.*



### 5.45.1 Detailed Description

Device queue.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.45.2 Function Documentation

#### 5.45.2.1 add\_dedicated\_queues()

```
bool lava::add_dedicated_queues (
    queue_family_info::list & list,
    VkQueueFamilyPropertiesList const & properties,
    r32 priority = 1.f)
```

Add dedicated queues.

#### Parameters

<i>list</i>	List of queue family informations
<i>properties</i>	List of queue family properties
<i>priority</i>	Queue priority

#### Returns

Add was successful or failed

#### 5.45.2.2 add\_queues()

```
bool lava::add_queues (
    queue_family_info::list & list,
    VkQueueFamilyPropertiesList const & properties,
    VkQueueFlags flags,
    ui32 count,
    r32 priority = 1.f)
```

Add queues.

#### Parameters

<i>list</i>	List of queue family informations
<i>properties</i>	List of queue family properties
<i>flags</i>	Queue flags
<i>count</i>	Number of queues
<i>priority</i>	Queue priority

#### Returns

Add was successful or failed

### 5.45.2.3 set\_all\_queues()

```
void lava::set_all_queues (
    queue_family_info::list & list,
    VkQueueFamilyPropertiesList const & properties)
```

Set all queues.

#### Parameters

<i>list</i>	List of queue family informations
<i>properties</i>	List of queue family properties

### 5.45.2.4 set\_default\_queues()

```
void lava::set_default_queues (
    queue_family_info::list & list)
```

Set the default queues.

#### Parameters

<i>list</i>	List of queue family informations
-------------	-----------------------------------

### 5.45.2.5 verify\_queues()

```
verify_queues_result lava::verify_queues (
    queue_family_info::list const & list,
    VkQueueFamilyPropertiesList const & properties)
```

Verify queues.

#### Parameters

<i>list</i>	List of queue family informations
<i>properties</i>	List of queue family properties

#### Returns

verify\_queues\_result Verification result

## 5.45.3 Variable Documentation

### 5.45.3.1 default\_queue\_flags

```
VkQueueFlags const lava::default_queue_flags [constexpr]
```

#### Initial value:

```
= VK_QUEUE_GRAPHICS_BIT
                                     | VK_QUEUE_COMPUTE_BIT
                                     | VK_QUEUE_TRANSFER_BIT
```

Default queue flags.

## 5.46 queue.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/base.hpp"
00011 #include <deque>
00012
00013 namespace lava {
00014
00018 struct queue {
00020     using list = std::deque<queue>;
00021
00023     using ref = queue const&;
00024
00026     VkQueue vk_queue = nullptr;
00027
00029     VkQueueFlags flags = 0;
00030
00032     index family = 0;
00033
00035     r32 priority = 1.f;
00036
00041     bool valid() const {
00042         return vk_queue != nullptr;
00043     }
00044
00050     bool operator<(queue const& other) const {
00051         return priority < other.priority;
00052     }
00053 };
00054
00056 constexpr VkQueueFlags const default_queue_flags = VK_QUEUE_GRAPHICS_BIT
00057                                                     | VK_QUEUE_COMPUTE_BIT
00058                                                     | VK_QUEUE_TRANSFER_BIT;
00059
00063 struct queue_info {
00065     using list = std::deque<queue_info>;
00066
00068     VkQueueFlags flags = default_queue_flags;
00069
00071     r32 priority = 1.f;
00072 };
00073
00077 struct queue_family_info {
00079     using list = std::deque<queue_family_info>;
00080
00082     index family_index = 0;
00083
00085     queue_info::list queues;
00086
00093     void add(VkQueueFlags flags,
00094             ui32 count = 1,
00095             r32 priority = 1.f) {
00096         for (auto i = 0u; i < count; ++i) {
00097             queue_info info{flags, priority};
00098             queues.push_back(info);
00099         }
00100     }
00101
00106     ui32 count() const {
00107         return to_ui32(queues.size());
00108     }
00109
00113     void clear() {
00114         queues.clear();
00115     }
00116 };
00117
00122 void set_default_queues(queue_family_info::list& list);
00123
00129 void set_all_queues(queue_family_info::list& list,
00130                     VkQueueFamilyPropertiesList const& properties);
00131
00141 bool add_queues(queue_family_info::list& list,
00142                 VkQueueFamilyPropertiesList const& properties,
00143                 VkQueueFlags flags,
00144                 ui32 count,
00145                 r32 priority = 1.f);
00146
00154 bool add_dedicated_queues(queue_family_info::list& list,
00155                           VkQueueFamilyPropertiesList const& properties,
00156                           r32 priority = 1.f);
00157

```

```

00161 enum class verify_queues_result : index {
00162     ok = 0,
00163     empty_list,
00164     no_properties,
00165     duplicate_family_index,
00166     no_family_index,
00167     no_queues,
00168     too_many_queues,
00169     no_compatible_flags
00170 };
00171
00178 verify_queues_result verify_queues(queue_family_info::list const& list,
00179                                     VkQueueFamilyPropertiesList const& properties);
00180
00181 } // namespace lava

```

## 5.47 liblava/block.hpp File Reference

Block module.

```

#include "liblava/block/attachment.hpp"
#include "liblava/block/block.hpp"
#include "liblava/block/compute_pipeline.hpp"
#include "liblava/block/def.hpp"
#include "liblava/block/descriptor.hpp"
#include "liblava/block/pipeline.hpp"
#include "liblava/block/pipeline_layout.hpp"
#include "liblava/block/render_pass.hpp"
#include "liblava/block/render_pipeline.hpp"
#include "liblava/block/subpass.hpp"

```

### 5.47.1 Detailed Description

Block module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.48 block.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/attachment.hpp"
00011 #include "liblava/block/block.hpp"
00012 #include "liblava/block/compute_pipeline.hpp"
00013 #include "liblava/block/def.hpp"
00014 #include "liblava/block/descriptor.hpp"
00015 #include "liblava/block/pipeline.hpp"
00016 #include "liblava/block/pipeline_layout.hpp"
00017 #include "liblava/block/render_pass.hpp"
00018 #include "liblava/block/render_pipeline.hpp"
00019 #include "liblava/block/subpass.hpp"

```

## 5.49 liblava/block/block.hpp File Reference

Command buffer model.

```
#include "liblava/base/device.hpp"
```

### Classes

- struct [lava::command](#)  
*Block command.*
- struct [lava::block](#)  
*Block of commands.*

### 5.49.1 Detailed Description

Command buffer model.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.50 block.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct command : entity {
00019     using s_ptr = std::shared_ptr<command>;
00020
00022     using c_ptr = command const*;
00023
00025     using s_map = std::map<id, s_ptr>;
00026
00028     using c_list = std::vector<c_ptr>;
00029
00031     VkCommandBuffers buffers = {};
00032
00034     using process_func = std::function<void(VkCommandBuffer)>;
00035
00037     process_func on_process;
00038
00040     bool active = true;
00041
00046     static s_ptr make() {
00047         return std::make_shared<command>();
00048     }
00049
00057     bool create(device::ptr device,
00058               index frame_count,
00059               VkCommandPools command_pools);
00060
```

```

00066     void destroy(device::ptr device,
00067                  VkCommandPools command_pools);
00068 };
00069
00073 struct block : entity {
00074     using ptr = block*;
00075
00076     using s_ptr = std::shared_ptr<block>;
00077
00078     using c_ptr = block const*;
00079
00080     using s_map = std::map<id, s_ptr>;
00081
00082     using c_list = std::vector<c_ptr>;
00083
00084     static s_ptr make() {
00085         return std::make_shared<block>();
00086     }
00087
00088     ~block() {
00089         destroy();
00090     }
00091
00092     bool create(device::ptr device,
00093                index frame_count,
00094                index queue_family);
00095
00096     void destroy();
00097
00098     index get_frame_count() const {
00099         return to_index(m_cmd_pools.size());
00100     }
00101
00102     id add_cmd(command::process_func func, bool active = true);
00103
00104     id add_command(command::process_func func, bool active = true) {
00105         return add_cmd(func, active);
00106     }
00107
00108     void remove_cmd(id::ref cmd_id);
00109
00110     void remove_command(id::ref cmd_id) {
00111         remove_cmd(cmd_id);
00112     }
00113
00114     bool process(index frame);
00115
00116     index get_current_frame() const {
00117         return m_current_frame;
00118     }
00119
00120     VkCommandBuffer get_command_buffer(id::ref cmd_id) const {
00121         return m_commands.at(cmd_id)->buffers.at(m_current_frame);
00122     }
00123
00124     VkCommandBuffer get_command_buffer(id::ref cmd_id, index frame) const {
00125         return m_commands.at(cmd_id)->buffers.at(frame);
00126     }
00127
00128     VkCommandBuffers collect_buffers() {
00129         VkCommandBuffers result;
00130
00131         for (auto& cmd : m_cmd_order)
00132             if (cmd->active)
00133                 result.push_back(cmd->buffers.at(m_current_frame));
00134
00135         return result;
00136     }
00137
00138     command::s_map const& get_commands() const {
00139         return m_commands;
00140     }
00141
00142     command::c_list const& get_cmd_order() const {
00143         return m_cmd_order;
00144     }
00145
00146     bool activated(id::ref cmd_id);
00147
00148     bool set_active(id::ref cmd_id, bool active = true);
00149
00150     device::ptr get_device() {
00151         return m_device;
00152     }
00153
00154 private:
00155     device::ptr m_device = nullptr;

```

```

00246
00248     index m_current_frame = 0;
00249
00251     VkCommandPools m_cmd_pools = {};
00252
00254     command::s_map m_commands;
00255
00257     command::c_list m_cmd_order;
00258 };
00259
00260 } // namespace lava

```

## 5.51 liblava/block/attachment.hpp File Reference

Attachment description.

```

#include "liblava/base/base.hpp"
#include <memory>

```

### Classes

- struct [lava::attachment](#)  
*Attachment description.*

### 5.51.1 Detailed Description

Attachment description.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.52 attachment.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/base.hpp"
00011 #include <memory>
00012
00013 namespace lava {
00014
00018 struct attachment {
00020     using s_ptr = std::shared_ptr<attachment>;
00021
00023     using s_list = std::vector<s_ptr>;
00024
00031     static s_ptr make(VkFormat format = VK_FORMAT_UNDEFINED,
00032                      VkSampleCountFlagBits samples = VK_SAMPLE_COUNT_1_BIT) {
00033         return std::make_shared<attachment>(format, samples);
00034     }
00035
00041     explicit attachment(VkFormat format = VK_FORMAT_UNDEFINED,

```

```

00042         VkSampleCountFlagBits samples = VK_SAMPLE_COUNT_1_BIT) {
00043     m_description.flags = 0;
00044     m_description.format = format;
00045     m_description.samples = samples;
00046     m_description.loadOp = VK_ATTACHMENT_LOAD_OP_LOAD;
00047     m_description.storeOp = VK_ATTACHMENT_STORE_OP_STORE;
00048     m_description.stencilLoadOp = VK_ATTACHMENT_LOAD_OP_LOAD;
00049     m_description.stencilStoreOp = VK_ATTACHMENT_STORE_OP_STORE;
00050     m_description.initialLayout = VK_IMAGE_LAYOUT_UNDEFINED;
00051     m_description.finalLayout = VK_IMAGE_LAYOUT_UNDEFINED;
00052 }
00053
00058 VkAttachmentDescription const& get_description() const {
00059     return m_description;
00060 }
00061
00066 void set_format(VkFormat format) {
00067     m_description.format = format;
00068 }
00069
00074 void set_samples(VkSampleCountFlagBits samples) {
00075     m_description.samples = samples;
00076 }
00077
00083 void set_op(VkAttachmentLoadOp load_op, VkAttachmentStoreOp store_op) {
00084     set_load_op(load_op);
00085     set_store_op(store_op);
00086 }
00087
00092 void set_load_op(VkAttachmentLoadOp load_op) {
00093     m_description.loadOp = load_op;
00094 }
00095
00100 void set_store_op(VkAttachmentStoreOp store_op) {
00101     m_description.storeOp = store_op;
00102 }
00103
00109 void set_stencil_op(VkAttachmentLoadOp load_op,
00110                    VkAttachmentStoreOp store_op) {
00111     set_stencil_load_op(load_op);
00112     set_stencil_store_op(store_op);
00113 }
00114
00119 void set_stencil_load_op(VkAttachmentLoadOp load_op) {
00120     m_description.stencilLoadOp = load_op;
00121 }
00122
00127 void set_stencil_store_op(VkAttachmentStoreOp store_op) {
00128     m_description.stencilStoreOp = store_op;
00129 }
00130
00136 void set_layouts(VkImageLayout initial,
00137                 VkImageLayout final) {
00138     set_initial_layout(initial);
00139     set_final_layout(final);
00140 }
00141
00146 void set_initial_layout(VkImageLayout layout) {
00147     m_description.initialLayout = layout;
00148 }
00149
00154 void set_final_layout(VkImageLayout layout) {
00155     m_description.finalLayout = layout;
00156 }
00157
00158 private:
00160     VkAttachmentDescription m_description;
00161 };
00162
00163 } // namespace lava

```

## 5.53 liblava/block/compute\_pipeline.hpp File Reference

Compute pipeline.

```
#include "liblava/block/pipeline.hpp"
```



**Classes**

- struct [lava::compute\\_pipeline](#)  
*Compute pipeline.*

**5.53.1 Detailed Description**

Compute pipeline.

**Authors**

Lava Block OÜ and contributors

**Copyright**

Copyright (c) 2018-present, MIT License

**5.54 compute\_pipeline.hpp**

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/pipeline.hpp"
00011
00012 namespace lava {
00013
00017 struct compute_pipeline : pipeline {
00019     using s_ptr = std::shared_ptr<compute_pipeline>;
00020
00022     using s_map = std::map<id, s_ptr>;
00023
00025     using s_list = std::vector<s_ptr>;
00026
00033     static s_ptr make(device::ptr device,
00034                       VkPipelineCache pipeline_cache = 0) {
00035         return std::make_shared<compute_pipeline>(device, pipeline_cache);
00036     }
00037
00039     using pipeline::pipeline;
00040
00045     void bind(VkCommandBuffer cmdBuffer) override;
00046
00053     bool set_shader_stage(c_data::ref data,
00054                          VkShaderStageFlagBits stage);
00055
00060     void set(shader_stage::s_ptr const& stage) {
00061         m_shader_stage = stage;
00062     }
00063
00068     shader_stage::s_ptr const& get_shader_stage() const {
00069         return m_shader_stage;
00070     }
00071
00076     void copy_to(compute_pipeline* target) const;
00077
00082     void copy_from(s_ptr const& source) {
00083         source->copy_to(this);
00084     }
00085
00086 private:
00091     bool setup() override;
00092
00096     void teardown() override;
00097
00099     shader_stage::s_ptr m_shader_stage;
00100 };
00101
00102 } // namespace lava

```

## 5.55 liblava/block/descriptor.hpp File Reference

Descriptor definition.

```
#include "liblava/base/device.hpp"
```

### Classes

- struct [lava::descriptor](#)  
*Descriptor.*
- struct [lava::descriptor::binding](#)  
*Descriptor binding.*
- struct [lava::descriptor::pool](#)  
*Descriptor pool.*

### 5.55.1 Detailed Description

Descriptor definition.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.56 descriptor.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct descriptor : entity {
00021     struct binding {
00023         using s_ptr = std::shared_ptr<binding>;
00024
00026         using s_list = std::vector<s_ptr>;
00027
00033         static s_ptr make(index index) {
00034             auto result = std::make_shared<binding>();
00035             result->set(index);
00036             result->set_count(1);
00037             return result;
00038         }
00039
00043         explicit binding();
00044
00049         VkDescriptorSetLayoutBinding const& get() const {
00050             return m_vk_binding;
00051         }
00052
00057         void set(index index) {
00058             m_vk_binding.binding = index;
00059         }
00059
```

```

00060
00065     void set_type(VkDescriptorType descriptor_type) {
00066         m_vk_binding.descriptorType = descriptor_type;
00067     }
00068
00073     void set_count(ui32 descriptor_count) {
00074         m_vk_binding.descriptorCount = descriptor_count;
00075     }
00076
00081     void set_stage_flags(VkShaderStageFlags stage_flags) {
00082         m_vk_binding.stageFlags = stage_flags;
00083     }
00084
00089     void set_samplers(VkSampler const* immutable_samplers) {
00090         m_vk_binding.pImmutableSamplers = immutable_samplers;
00091     }
00092
00093 private:
00095     VkDescriptorSetLayoutBinding m_vk_binding;
00096 };
00097
00101 struct pool : entity {
00103     using s_ptr = std::shared_ptr<pool>;
00104
00106     using s_list = std::vector<s_ptr>;
00107
00112     static s_ptr make() {
00113         return std::make_shared<descriptor::pool>();
00114     }
00115
00124     bool create(device::ptr device,
00125                VkDescriptorPoolSizesRef sizes,
00126                ui32 max = 1,
00127                VkDescriptorPoolCreateFlags flags =
00128                    VK_DESCRIPTOR_POOL_CREATE_FREE_DESCRIPTOR_SET_BIT);
00129
00133     void destroy();
00134
00139     VkDescriptorPool get() const {
00140         return m_vk_pool;
00141     }
00142
00147     device::ptr get_device() {
00148         return m_device;
00149     }
00150
00155     VkDescriptorPoolSizes const& get_sizes() const {
00156         return m_sizes;
00157     }
00158
00163     ui32 get_max() const {
00164         return m_max;
00165     }
00166
00167 private:
00169     device::ptr m_device = nullptr;
00170
00172     VkDescriptorPool m_vk_pool = VK_NULL_HANDLE;
00173
00175     VkDescriptorPoolSizes m_sizes;
00176
00178     ui32 m_max = 0;
00179 };
00180
00182 using s_ptr = std::shared_ptr<descriptor>;
00183
00185 using s_list = std::vector<s_ptr>;
00186
00191 static s_ptr make() {
00192     return std::make_shared<descriptor>();
00193 }
00194
00201 void add_binding(index binding,
00202                 VkDescriptorType descriptor_type,
00203                 VkShaderStageFlags stage_flags);
00204
00208 void clear_bindings() {
00209     m_bindings.clear();
00210 }
00211
00216 void add(binding::s_ptr const& binding) {
00217     m_bindings.push_back(binding);
00218 }
00219
00225 bool create(device::ptr device);
00226
00230 void destroy();

```

```

00231
00236     ui32 get_binding_count() const {
00237         return to_ui32(m_bindings.size());
00238     }
00239
00244     binding::s_list const& get_bindings() {
00245         return m_bindings;
00246     }
00247
00252     VkDescriptorSetLayout get() const {
00253         return m_layout;
00254     }
00255
00260     device::ptr get_device() {
00261         return m_device;
00262     }
00263
00269     VkDescriptorSet allocate_set(VkDescriptorPool pool);
00270
00274     VkDescriptorSet allocate(VkDescriptorPool pool) {
00275         return allocate_set(pool);
00276     }
00277
00284     bool deallocate_set(VkDescriptorSet& descriptor_set,
00285                         VkDescriptorPool pool);
00286
00290     bool deallocate(VkDescriptorSet& descriptor_set,
00291                    VkDescriptorPool pool) {
00292         return deallocate_set(descriptor_set, pool);
00293     }
00294
00301     VkDescriptorSets allocate_sets(ui32 size,
00302                                   VkDescriptorPool pool);
00303
00307     VkDescriptorSets allocate(ui32 size,
00308                               VkDescriptorPool pool) {
00309         return allocate_sets(size, pool);
00310     }
00311
00318     bool deallocate_sets(VkDescriptorSets& descriptor_sets,
00319                          VkDescriptorPool pool);
00320
00324     bool deallocate(VkDescriptorSets& descriptor_sets,
00325                     VkDescriptorPool pool) {
00326         return deallocate_sets(descriptor_sets, pool);
00327     }
00328
00329 private:
00331     device::ptr m_device = nullptr;
00332
00334     VkDescriptorSetLayout m_layout = VK_NULL_HANDLE;
00335
00337     binding::s_list m_bindings;
00338 };
00339
00340 } // namespace lava

```

## 5.57 liblava/block/pipeline.hpp File Reference

Pipeline.

```
#include "liblava/block/pipeline_layout.hpp"
```

### Classes

- struct [lava::pipeline](#)  
*Pipeline.*
- struct [lava::pipeline::shader\\_stage](#)  
*Shader stage.*

## Functions

- [pipeline::shader\\_stage::s\\_ptr](#) [lava::create\\_pipeline\\_shader\\_stage](#) ([device::ptr](#) device, [c\\_data::ref](#) data, [VkShaderStageFlagBits](#) stage)

*Create a new pipeline shader stage.*

### 5.57.1 Detailed Description

Pipeline.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.57.2 Function Documentation

#### 5.57.2.1 create\_pipeline\_shader\_stage()

```
pipeline::shader_stage::s_ptr lava::create_pipeline_shader_stage (  
    device::ptr device,  
    c_data::ref data,  
    VkShaderStageFlagBits stage)
```

Create a new pipeline shader stage.

#### Parameters

<i>device</i>	Vulkan device
<i>data</i>	Shader data
<i>stage</i>	Shader stage flag bits

#### Returns

[pipeline::shader\\_stage::s\\_ptr](#) Shared pointer to pipeline shader stage

## 5.58 pipeline.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/pipeline_layout.hpp"
00011
00012 namespace lava {
00013
00017 struct pipeline : entity {
00019     using s_ptr = std::shared_ptr<pipeline>;
00020
00022     using s_list = std::vector<s_ptr>;
00023
00025     using process_func = std::function<void(VkCommandBuffer)>;
00026
00028     process_func on_process;
00029
00035     explicit pipeline(device::ptr device,
00036                     VkPipelineCache pipeline_cache = 0);
00037
00041     ~pipeline() override;
00042
00047     bool create();
00048
00052     void destroy();
00053
00058     virtual void bind(VkCommandBuffer cmd_buf) = 0;
00059
00064     void set_active(bool value = true) {
00065         m_active = value;
00066     }
00067
00072     bool activated() const {
00073         return m_active;
00074     }
00075
00079     void toggle() {
00080         m_active = !m_active;
00081     }
00082
00087     void set_auto_bind(bool value = true) {
00088         m_auto_bind_active = value;
00089     }
00090
00095     bool auto_bind() const {
00096         return m_auto_bind_active;
00097     }
00098
00103     bool ready() const {
00104         return m_vk_pipeline != VK_NULL_HANDLE;
00105     }
00106
00111     VkPipeline get() const {
00112         return m_vk_pipeline;
00113     }
00114
00119     device::ptr get_device() {
00120         return m_device;
00121     }
00122
00127     pipeline_layout::s_ptr get_layout() const {
00128         return m_layout;
00129     }
00130
00135     void set_layout(pipeline_layout::s_ptr const& value) {
00136         m_layout = value;
00137     }
00138
00142     struct shader_stage {
00144         using s_ptr = std::shared_ptr<shader_stage>;
00145
00147         using s_list = std::vector<s_ptr>;
00148
00154         static s_ptr make(VkShaderStageFlagBits stage) {
00155             auto result = std::make_shared<shader_stage>();
00156             result->set_stage(stage);
00157             return result;
00158         }
00159
00163         explicit shader_stage();
00164
00168         ~shader_stage();
00169

```

```

00174     void set_stage(VkShaderStageFlagBits stage) {
00175         m_create_info.stage = stage;
00176     }
00177
00182     void add_specialization_entry(VkSpecializationMapEntry const& specialization);
00183
00191     bool create(device::ptr device,
00192                c_data::ref shader_data,
00193                c_data::ref specialization_data = data());
00194
00198     void destroy();
00199
00204     VkPipelineShaderStageCreateInfo const& get_create_info() const {
00205         return m_create_info;
00206     }
00207
00208 private:
00210     device::ptr m_device = nullptr;
00211
00213     VkPipelineShaderStageCreateInfo m_create_info;
00214
00216     VkSpecializationInfo m_specialization_info;
00217
00219     VkSpecializationMapEntries m_specialization_entries;
00220
00222     data m_specialization_data_copy;
00223 };
00224
00225 protected:
00230     virtual bool setup() = 0;
00231
00235     virtual void teardown() = 0;
00236
00238     device::ptr m_device = nullptr;
00239
00241     VkPipeline m_vk_pipeline = VK_NULL_HANDLE;
00242
00244     VkPipelineCache m_pipeline_cache = VK_NULL_HANDLE;
00245
00247     pipeline_layout::s_ptr m_layout;
00248
00249 private:
00251     bool m_active = true;
00252
00254     bool m_auto_bind_active = true;
00255 };
00256
00264 pipeline::shader_stage::s_ptr create_pipeline_shader_stage(device::ptr device,
00265                                                            c_data::ref data,
00266                                                            VkShaderStageFlagBits stage);
00267
00268 } // namespace lava

```

## 5.59 liblava/block/pipeline\_layout.hpp File Reference

Pipeline layout.

```
#include "liblava/block/descriptor.hpp"
```

### Classes

- struct [lava::pipeline\\_layout](#)  
*Pipeline layout.*

### 5.59.1 Detailed Description

Pipeline layout.

## Authors

Lava Block OÜ and contributors

## Copyright

Copyright (c) 2018-present, MIT License

## 5.60 pipeline\_layout.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/descriptor.hpp"
00011
00012 namespace lava {
00013
00017 struct pipeline_layout : entity {
00019     using s_ptr = std::shared_ptr<pipeline_layout>;
00020
00022     using s_list = std::vector<s_ptr>;
00023
00028     static s_ptr make() {
00029         return std::make_shared<pipeline_layout>();
00030     }
00031
00035     void add(descriptor::s_ptr const& descriptor) {
00036         m_descriptors.push_back(descriptor);
00037     }
00038
00042     void add(VkPushConstantRange const& range) {
00043         m_push_constant_ranges.push_back(range);
00044     }
00045
00050     void add_descriptor(descriptor::s_ptr const& descriptor) {
00051         add(descriptor);
00052     }
00053
00058     void add_push_constant_range(VkPushConstantRange const& range) {
00059         add(range);
00060     }
00061
00065     void clear_descriptors() {
00066         m_descriptors.clear();
00067     }
00068
00072     void clear_ranges() {
00073         m_push_constant_ranges.clear();
00074     }
00075
00079     void clear() {
00080         clear_descriptors();
00081         clear_ranges();
00082     }
00083
00089     bool create(device::ptr device);
00090
00094     void destroy();
00095
00100     VkPipelineLayout get() const {
00101         return m_layout;
00102     }
00103
00108     device::ptr get_device() {
00109         return m_device;
00110     }
00111
00116     descriptor::s_list const& get_descriptors() const {
00117         return m_descriptors;
00118     }
00119
00124     VkPushConstantRanges const& get_push_constant_ranges() const {
00125         return m_push_constant_ranges;
00126     }
00127
00129     using offset_list = std::vector<index>;
00130

```



```

00139     void bind_descriptor_set(VkCommandBuffer cmd_buf,
00140                             VkDescriptorSet descriptor_set,
00141                             index first_set = 0,
00142                             offset_list offsets = {},
00143                             VkPipelineBindPoint bind_point = VK_PIPELINE_BIND_POINT_GRAPHICS);
00144
00148     void bind(VkCommandBuffer cmd_buf,
00149              VkDescriptorSet descriptor_set,
00150              index first_set = 0,
00151              offset_list offsets = {},
00152              VkPipelineBindPoint bind_point = VK_PIPELINE_BIND_POINT_GRAPHICS) {
00153         bind_descriptor_set(cmd_buf,
00154                             descriptor_set,
00155                             first_set,
00156                             offsets,
00157                             bind_point);
00158     }
00159
00160 private:
00162     device::ptr m_device = nullptr;
00163
00165     VkPipelineLayout m_layout = VK_NULL_HANDLE;
00166
00168     descriptor::s_list m_descriptors;
00169
00171     VkPushConstantRanges m_push_constant_ranges;
00172 };
00173
00174 } // namespace lava

```

## 5.61 liblava/block/render\_pass.hpp File Reference

Render pass.

```

#include "liblava/base/device.hpp"
#include "liblava/block/attachment.hpp"
#include "liblava/block/subpass.hpp"

```

### Classes

- struct [lava::render\\_pass](#)  
*Render pass.*

### 5.61.1 Detailed Description

Render pass.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.62 render\_pass.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011 #include "liblava/block/attachment.hpp"
00012 #include "liblava/block/subpass.hpp"
00013
00014 namespace lava {
00015
00019 struct render_pass : entity {
00021     using s_ptr = std::shared_ptr<render_pass>;
00022
00024     using s_list = std::vector<s_ptr>;
00025
00031     static s_ptr make(device::ptr device) {
00032         return std::make_shared<render_pass>(device);
00033     }
00034
00039     explicit render_pass(device::ptr device);
00040
00047     bool create(VkAttachmentsRef target_attachments,
00048               rect::ref area);
00049
00053     void destroy();
00054
00060     void process(VkCommandBuffer cmd_buf,
00061               index frame);
00062
00067     device::ptr get_device() {
00068         return m_device;
00069     }
00070
00075     VkRenderPass get() const {
00076         return m_vk_render_pass;
00077     }
00078
00083     ui32 get_subpass_count() const {
00084         return to_ui32(m_subpasses.size());
00085     }
00086
00092     bool exists_subpass(index index = 0) const {
00093         return index < m_subpasses.size();
00094     }
00095
00101     subpass* get_subpass(index index = 0) {
00102         return m_subpasses.at(index).get();
00103     }
00104
00109     subpass::s_list const& get_subpasses() const {
00110         return m_subpasses;
00111     }
00112
00117     void add(attachment::s_ptr const& attachment) {
00118         m_attachments.push_back(attachment);
00119     }
00120
00125     void add(subpass_dependency::s_ptr const& dependency) {
00126         m_dependencies.push_back(dependency);
00127     }
00128
00133     void add(subpass::s_ptr const& subpass) {
00134         m_subpasses.push_back(subpass);
00135     }
00136
00141     void set_clear_values(VkClearValues const& values) {
00142         m_clear_values = values;
00143     }
00144
00149     VkClearValues const& get_clear_values() const {
00150         return m_clear_values;
00151     }
00152
00157     void set_clear_color(v3 value = {});
00158
00163     v3 get_clear_color() const;
00164
00170     void add(render_pipeline::s_ptr pipeline,
00171             index subpass = 0) {
00172         m_subpasses.at(subpass)->add(pipeline);
00173     }
00174
00180     void add_front(render_pipeline::s_ptr pipeline,

```

```

00181         index subpass = 0) {
00182             m_subpasses.at(subpass)->add_front(pipeline);
00183         }
00184
00190     void remove(render_pipeline::s_ptr pipeline,
00191                 index subpass = 0) {
00192         m_subpasses.at(subpass)->remove(pipeline);
00193     }
00194
00199     target_callback const& get_target_callback() const {
00200         return m_callback;
00201     }
00202
00203 private:
00205     device::ptr m_device = nullptr;
00206
00208     VkRenderPass m_vk_render_pass = VK_NULL_HANDLE;
00209
00211     VkFramebuffers m_framebuffers = {};
00212
00214     attachment::s_list m_attachments;
00215
00217     subpass_dependency::s_list m_dependencies;
00218
00220     subpass::s_list m_subpasses;
00221
00223     VkClearValues m_clear_values = {};
00224
00226     rect m_area;
00227
00229     target_callback m_callback;
00230
00236     void begin(VkCommandBuffer cmd_buf,
00237               index frame);
00238
00243     void end(VkCommandBuffer cmd_buf);
00244
00251     bool on_target_created(VkAttachmentsRef target_attachments,
00252                           rect::ref area);
00253
00257     void on_target_destroyed();
00258 };
00259
00260 } // namespace lava

```

## 5.63 liblava/block/render\_pipeline.hpp File Reference

Render pipeline (Graphics)

```
#include "liblava/block/pipeline.hpp"
```

### Classes

- struct [lava::render\\_pipeline](#)  
*Render pipeline (Graphics)*
- struct [lava::render\\_pipeline::create\\_info](#)  
*Render pipeline create information.*

### Functions

- VkPipelineColorBlendAttachmentState [lava::create\\_pipeline\\_color\\_blend\\_attachment\(\)](#)  
*Create a color blend attachment.*

### 5.63.1 Detailed Description

Render pipeline (Graphics)

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.63.2 Function Documentation

#### 5.63.2.1 create\_pipeline\_color\_blend\_attachment()

```
VkPipelineColorBlendAttachmentState lava::create_pipeline_color_blend_attachment ()
```

Create a color blend attachment.

#### Returns

VkPipelineColorBlendAttachmentState Pipeline color blend attachment state

## 5.64 render\_pipeline.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/block/pipeline.hpp"
00011
00012 namespace lava {
00013
00017 struct render_pipeline : pipeline {
00019     using s_ptr = std::shared_ptr<render_pipeline>;
00020
00022     using s_map = std::map<id, s_ptr>;
00023
00025     using s_list = std::vector<s_ptr>;
00026
00030     enum class sizing_mode : index {
00031         input = 0,
00032         absolute,
00033         relative
00034     };
00035
00039     struct create_info {
00041         VkPipelineVertexInputStateCreateInfo vertex_input_state;
00042
00044         VkPipelineInputAssemblyStateCreateInfo input_assembly_state;
00045
00047         VkPipelineViewportStateCreateInfo viewport_state;
00048
00050         VkPipelineMultisampleStateCreateInfo multisample_state;
00051
00053         VkPipelineDepthStencilStateCreateInfo depth_stencil_state;
00054
00056         VkPipelineRasterizationStateCreateInfo rasterization_state;
00057     };
00058
00065     static s_ptr make(device::ptr device,
00066                       VkPipelineCache pipeline_cache = 0) {
00067         return std::make_shared<render_pipeline>(device, pipeline_cache);
00068     }
00069 }
```

```

00069
00075     explicit render_pipeline(device::ptr device,
00076                             VkPipelineCache pipeline_cache);
00077
00082     void bind(VkCommandBuffer cmd_buf) override;
00083
00089     void set_viewport_and_scissor(VkCommandBuffer cmd_buf,
00090                                   uv2 size);
00091
00096     void set_render_pass(VkRenderPass pass) {
00097         m_render_pass = pass;
00098     }
00099
00103     void set(VkRenderPass pass) {
00104         set_render_pass(pass);
00105     }
00106
00111     VkRenderPass get_render_pass() const {
00112         return m_render_pass;
00113     }
00114
00119     index get_subpass() const {
00120         return m_subpass;
00121     }
00122
00127     void set_subpass(index value) {
00128         m_subpass = value;
00129     }
00130
00136     bool create(VkRenderPass pass) {
00137         set(pass);
00138
00139         return pipeline::create();
00140     }
00141
00146     void set_vertex_input_binding(VkVertexInputBindingDescription const& description);
00147
00152     void set_vertex_input_bindings(VkVertexInputBindingDescriptions const& descriptions);
00153
00158     void set_vertex_input_attribute(VkVertexInputAttributeDescription const& attribute);
00159
00164     void set_vertex_input_attributes(VkVertexInputAttributeDescriptions const& attributes);
00165
00170     void set_input_topology(VkPrimitiveTopology const& topology);
00171
00177     void set_depth_test_and_write(bool test_enable = true,
00178                                   bool write_enable = true);
00179
00184     void set_depth_compare_op(VkCompareOp compare_op);
00185
00190     void set_rasterization_cull_mode(VkCullModeFlags cull_mode);
00191
00196     void set_rasterization_front_face(VkFrontFace front_face);
00197
00202     void set_rasterization_polygon_mode(VkPolygonMode polygon_mode);
00203
00208     void add_color_blend_attachment(VkPipelineColorBlendAttachmentState const& attachment);
00209
00213     void add_color_blend_attachment();
00214
00218     void clear_color_blend_attachment();
00219
00224     void set_dynamic_states(VkDynamicStates const& states);
00225
00230     void add_dynamic_state(VkDynamicState state);
00231
00235     void clear_dynamic_states();
00236
00243     bool add_shader_stage(c_data::ref data,
00244                           VkShaderStageFlagBits stage);
00245
00252     bool add_shader(c_data::ref data,
00253                     VkShaderStageFlagBits stage) {
00254         return add_shader_stage(data, stage);
00255     }
00256
00261     void add(shader_stage::s_ptr const& shader_stage) {
00262         m_shader_stages.push_back(shader_stage);
00263     }
00264
00269     shader_stage::s_list const& get_shader_stages() const {
00270         return m_shader_stages;
00271     }
00272
00276     void clear_shader_stages() {
00277         m_shader_stages.clear();
00278     }

```

```

00279
00283 void clear() {
00284     clear_color_blend_attachment();
00285     clear_shader_stages();
00286 }
00287
00292 void set_auto_size(bool value = true) {
00293     m_auto_size = value;
00294 }
00295
00300 bool auto_sizing() const {
00301     return m_auto_size;
00302 }
00303
00308 VkViewport get_viewport() const {
00309     return m_viewport;
00310 }
00311
00316 void set_viewport(VkViewport value) {
00317     m_viewport = value;
00318 }
00319
00324 VkRect2D get_scissor() const {
00325     return m_scissor;
00326 }
00327
00332 void set_scissor(VkRect2D value) {
00333     m_scissor = value;
00334 }
00335
00340 sizing_mode get_sizing() const {
00341     return m_sizing;
00342 }
00343
00348 void set_sizing(sizing_mode value) {
00349     m_sizing = value;
00350 }
00351
00356 void copy_to(render_pipeline* target) const;
00357
00362 void copy_from(s_ptr const& source) {
00363     source->copy_to(this);
00364 }
00365
00370 void set_line_width(r32 value) {
00371     m_line_width = value;
00372 }
00373
00378 r32 get_line_width() const {
00379     return m_line_width;
00380 }
00381
00386 bool auto_line_width() const {
00387     return m_auto_line_width_state;
00388 }
00389
00394 void set_auto_line_width(bool value = true) {
00395     m_auto_line_width_state = value;
00396 }
00397
00402 void set_line_width(VkCommandBuffer cmd_buf) {
00403     vkCmdSetLineWidth(cmd_buf, m_line_width);
00404 }
00405
00407 using create_func = std::function<bool(create_info&)>;
00408
00410 create_func on_create;
00411
00412 private:
00417 bool setup() override;
00418
00422 void teardown() override;
00423
00425 VkRenderPass m_render_pass = VK_NULL_HANDLE;
00426
00428 index m_subpass = 0;
00429
00431 create_info m_info;
00432
00434 VkVertexInputBindingDescriptions m_vertex_input_bindings;
00435
00437 VkVertexInputAttributeDescriptions m_vertex_input_attributes;
00438
00440 VkPipelineColorBlendAttachmentStates m_color_blend_attachment_states;
00441
00443 VkPipelineColorBlendStateCreateInfo m_color_blend_state;
00444

```

```

00446     VkPipelineDynamicStateCreateInfo m_dynamic_state;
00447
00449     VkDynamicStates m_dynamic_states;
00450
00452     shader_stage::s_list m_shader_stages;
00453
00455     sizing\_mode m_sizing = sizing_mode::input;
00456
00458     VkViewport m_viewport;
00459
00461     VkRect2D m_scissor;
00462
00464     bool m_auto_size = true;
00465
00467     bool m_auto_line_width_state = false;
00468
00470     r32 m_line_width = 1.f;
00471 };
00472
00477 VkPipelineColorBlendAttachmentState create\_pipeline\_color\_blend\_attachment();
00478
00479 } // namespace lava

```

## 5.65 liblava/block/subpass.hpp File Reference

Subpass.

```
#include "liblava/block/render_pipeline.hpp"
```

### Classes

- struct [lava::subpass](#)  
*Subpass.*
- struct [lava::subpass\\_dependency](#)  
*Subpass dependency.*

### 5.65.1 Detailed Description

Subpass.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.66 subpass.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/block/render_pipeline.hpp"
00011
00012 namespace lava {
00013
00017 struct subpass : entity {
00019     using s_ptr = std::shared_ptr<subpass>;
00020
00022     using s_list = std::vector<s_ptr>;
00023
00029     static s_ptr make(VkPipelineBindPoint pipeline_bind_point =
00030                       VK_PIPELINE_BIND_POINT_GRAPHICS) {
00031         auto result = std::make_shared<subpass>();
00032         result->set(pipeline_bind_point);
00033         return result;
00034     }
00035
00039     explicit subpass();
00040
00044     void destroy();
00045
00050     void add(render_pipeline::s_ptr const& pipeline) {
00051         m_pipelines.push_back(pipeline);
00052     }
00053
00058     void add_front(render_pipeline::s_ptr const& pipeline) {
00059         m_pipelines.insert(m_pipelines.begin(), pipeline);
00060     }
00061
00066     void remove(render_pipeline::s_ptr pipeline);
00067
00071     void clear_pipelines();
00072
00078     void process(VkCommandBuffer cmd_buf,
00079                 uv2 size);
00080
00085     VkSubpassDescription const& get_description() const {
00086         return m_description;
00087     }
00088
00093     void set(VkPipelineBindPoint pipeline_bind_point) {
00094         m_description.pipelineBindPoint = pipeline_bind_point;
00095     }
00096
00102     void set_color_attachment(index attachment,
00103                              VkImageLayout layout);
00104
00109     void set_color_attachment(VkAttachmentReference attachment);
00110
00115     void set_color_attachments(VkAttachmentReferences const& attachments);
00116
00122     void set_depth_stencil_attachment(index attachment,
00123                                       VkImageLayout layout);
00124
00129     void set_depth_stencil_attachment(VkAttachmentReference attachment);
00130
00136     void set_input_attachment(index attachment,
00137                               VkImageLayout layout);
00138
00143     void set_input_attachment(VkAttachmentReference attachment);
00144
00149     void set_input_attachments(VkAttachmentReferences const& attachments);
00150
00156     void set_resolve_attachment(index attachment, VkImageLayout layout);
00157
00162     void set_resolve_attachment(VkAttachmentReference attachment);
00163
00168     void set_resolve_attachments(VkAttachmentReferences const& attachments);
00169
00174     void add_preserve_attachment(index attachment);
00175
00180     void set_preserve_attachments(index_list const& attachments);
00181
00186     void set_active(bool value = true) {
00187         m_active = value;
00188     }
00189
00194     bool activated() const {
00195         return m_active;
00196     }

```



```

00197
00198 private:
00200     VkSubpassDescription m_description;
00201
00202     bool m_active = true;
00203
00204     VkAttachmentReferences m_color_attachments;
00205
00206     VkAttachmentReference m_depth_stencil_attachment{};
00207
00208     VkAttachmentReferences m_input_attachments;
00209
00210     VkAttachmentReferences m_resolve_attachments;
00211
00212     index_list m_preserve_attachments;
00213
00214     render_pipeline::s_list m_pipelines;
00215 };
00216
00217 struct subpass_dependency {
00218     using s_ptr = std::shared_ptr<subpass_dependency>;
00219
00220     using s_list = std::vector<s_ptr>;
00221
00222     static s_ptr make(ui32 src_subpass,
00223                       ui32 dst_subpass,
00224                       VkDependencyFlags dependency_flags =
00225                           VK_DEPENDENCY_BY_REGION_BIT) {
00226         auto result = std::make_shared<subpass_dependency>();
00227         result->set_subpass(src_subpass, dst_subpass);
00228         result->set_dependency_flags(dependency_flags);
00229         return result;
00230     }
00231
00232     explicit subpass_dependency();
00233
00234     VkSubpassDependency const& get_dependency() const {
00235         return m_dependency;
00236     }
00237
00238     void set_subpass(ui32 src, ui32 dst) {
00239         set_src_subpass(src);
00240         set_dst_subpass(dst);
00241     }
00242
00243     void set_src_subpass(ui32 src) {
00244         m_dependency.srcSubpass = src;
00245     }
00246
00247     void set_dst_subpass(ui32 dst) {
00248         m_dependency.dstSubpass = dst;
00249     }
00250
00251     void set_stage_mask(VkPipelineStageFlags src,
00252                        VkPipelineStageFlags dst) {
00253         set_src_stage_mask(src);
00254         set_dst_stage_mask(dst);
00255     }
00256
00257     void set_src_stage_mask(VkPipelineStageFlags mask) {
00258         m_dependency.srcStageMask = mask;
00259     }
00260
00261     void set_dst_stage_mask(VkPipelineStageFlags mask) {
00262         m_dependency.dstStageMask = mask;
00263     }
00264
00265     void set_access_mask(VkAccessFlags src,
00266                        VkAccessFlags dst) {
00267         set_src_access_mask(src);
00268         set_dst_access_mask(dst);
00269     }
00270
00271     void set_src_access_mask(VkAccessFlags mask) {
00272         m_dependency.srcAccessMask = mask;
00273     }
00274
00275     void set_dst_access_mask(VkAccessFlags mask) {
00276         m_dependency.dstAccessMask = mask;
00277     }
00278
00279     void set_dependency_flags(VkDependencyFlags flags) {
00280         m_dependency.dependencyFlags = flags;
00281     }
00282 private:
00283     VkSubpassDependency m_dependency;

```

```
00355 };  
00356  
00357 } // namespace lava
```

## 5.67 liblava/core.hpp File Reference

Core module.

```
#include "liblava/core/data.hpp"  
#include "liblava/core/def.hpp"  
#include "liblava/core/id.hpp"  
#include "liblava/core/misc.hpp"  
#include "liblava/core/time.hpp"  
#include "liblava/core/types.hpp"  
#include "liblava/core/version.hpp"
```

### 5.67.1 Detailed Description

Core module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.68 core.hpp

[Go to the documentation of this file.](#)

```
00001  
00008 #pragma once  
00009  
00010 #include "liblava/core/data.hpp"  
00011 #include "liblava/core/def.hpp"  
00012 #include "liblava/core/id.hpp"  
00013 #include "liblava/core/misc.hpp"  
00014 #include "liblava/core/time.hpp"  
00015 #include "liblava/core/types.hpp"  
00016 #include "liblava/core/version.hpp"
```

## 5.69 liblava/core/data.hpp File Reference

Data wrapper.

```
#include "liblava/core/types.hpp"  
#include <string.h>
```

## Classes

- struct [lava::data](#)  
*Data wrapper.*
- struct [lava::data\\_provider](#)  
*Data provider.*
- struct [lava::c\\_data](#)  
*Const data wrapper.*
- struct [lava::u\\_data](#)  
*Unique data wrapper.*

## Functions

- auto [lava::align\\_up](#) (auto value, auto [align](#))  
*Align value up.*
- [size\\_t lava::align](#) ([size\\_t](#) size, [size\\_t](#) min=0)  
*Align a size.*
- template<typename T >  
[size\\_t lava::align](#) ([size\\_t](#) min=0)  
*Get alignment of type.*
- void \* [lava::alloc\\_data](#) ([size\\_t](#) size, [size\\_t](#) alignment=sizeof(c8))  
*Allocate data.*
- void [lava::free\\_data](#) (void \*data)  
*Free data.*
- void \* [lava::realloc\\_data](#) (void \*data, [size\\_t](#) size, [size\\_t](#) alignment=sizeof(c8))  
*Reallocate data.*
- [size\\_t lava::next\\_pow\\_2](#) ([size\\_t](#) x)  
*Get next power of two.*

### 5.69.1 Detailed Description

Data wrapper.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.69.2 Function Documentation

#### 5.69.2.1 align() [1/2]

```
template<typename T >
size_t lava::align (
    size_t min = 0) [inline]
```

Get alignment of type.

### Template Parameters

<i>T</i>	Type to align
----------	---------------

### Parameters

<i>min</i>	Minimal alignment
------------	-------------------

### Returns

size\_t Aligned size

## 5.69.2.2 align() [2/2]

```
size_t lava::align (  
    size_t size,  
    size_t min = 0) [inline]
```

Align a size.

### Parameters

<i>size</i>	Site to align
<i>min</i>	Minimal alignment

### Returns

size\_t Aligned size

## 5.69.2.3 align\_up()

```
auto lava::align_up (  
    auto value,  
    auto align) [inline]
```

Align value up.

### Parameters

<i>value</i>	Value to align
<i>align</i>	Target alignment

### Returns

auto Aligned value

## 5.69.2.4 alloc\_data()

```
void * lava::alloc_data (  
    size_t size,  
    size_t alignment = sizeof(c8)) [inline]
```

Allocate data.

## Parameters

<i>size</i>	Size of data
<i>alignment</i>	Target alignment

## Returns

void\* Allocated data

**5.69.2.5 free\_data()**

```
void lava::free_data (  
    void * data) [inline]
```

Free data.

## Parameters

<i>data</i>	Data to free
-------------	--------------

**5.69.2.6 next\_pow\_2()**

```
size_t lava::next_pow_2 (  
    size_t x) [inline]
```

Get next power of two.

## Parameters

<i>x</i>	Source value
----------	--------------

## Returns

size\_t Next power of two

**5.69.2.7 realloc\_data()**

```
void * lava::realloc_data (  
    void * data,  
    size_t size,  
    size_t alignment = sizeof(c8)) [inline]
```

Reallocate data.

## Parameters

<i>data</i>	Data to reallocate
<i>size</i>	Size of data
<i>alignment</i>	Target alignment

## Returns

void\* Reallocated data

## 5.70 data.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include <string.h>
00012
00013 namespace lava {
00014
00021 inline auto align_up(auto value,
00022                     auto align) {
00023     return (value + align - 1) / align * align;
00024 }
00025
00032 inline size_t align(size_t size,
00033                    size_t min = 0) {
00034     if (min == 0)
00035         return align_up(size, sizeof(void*));
00036
00037     return align_up((size + min - 1) & ~(min - 1), sizeof(void*));
00038 }
00039
00046 template <typename T>
00047 inline size_t align(size_t min = 0) {
00048     return align(sizeof(T), min);
00049 }
00050
00057 inline void* alloc_data(size_t size,
00058                        size_t alignment = sizeof(c8)) {
00059     #if _WIN32
00060         return _aligned_malloc(size, alignment);
00061     #else
00062         if (size % alignment == 0) {
00063             return aligned_alloc(alignment, size);
00064         } else {
00065             return aligned_alloc(alignment, ((size / alignment) + 1) * alignment);
00066         }
00067     #endif
00068 }
00069
00074 inline void free_data(void* data) {
00075     #if _WIN32
00076         _aligned_free(data);
00077     #else
00078         free(data);
00079     #endif
00080 }
00081
00089 inline void* realloc_data(void* data,
00090                          size_t size,
00091                          size_t alignment = sizeof(c8)) {
00092     #if _WIN32
00093         return _aligned_realloc(data, size, alignment);
00094     #else
00095         return realloc(data, align(size, alignment));
00096     #endif
00097 }
00098
00102 struct data {
00104     using ref = data const&;
00105
00107     using ptr = char*;
00108
00110     using c_ptr = char const*;
00111
00117     static inline ptr as_ptr(auto* value) {
00118         return (ptr)value;
00119     }
00120
00126     static inline c_ptr as_c_ptr(auto* value) {
00127         return (c_ptr)value;
00128     }
00129
00133     enum class mode : index {
00134         alloc = 0,
00135         no_alloc
00136     };
00137
00141     data() = default;
00142
00148     data(auto* addr, size_t size)
00149     : addr(as_ptr(addr)), size(size) {}
00150

```

```

00157     bool set(size_t length,
00158             mode mode = mode::alloc) {
00159         size = length;
00160         alignment = align<data::ptr>();
00161
00162         if (mode == mode::alloc)
00163             return allocate();
00164
00165         return true;
00166     }
00167
00172     bool allocate() {
00173         addr = as_ptr(alloc_data(size, alignment));
00174         return addr != nullptr;
00175     }
00176
00180     void deallocate() {
00181         if (!addr)
00182             return;
00183
00184         free_data(addr);
00185         addr = nullptr;
00186     }
00187
00192     ptr end() const {
00193         return addr + size;
00194     }
00195
00197     ptr addr = nullptr;
00198
00200     size_t size = 0;
00201
00203     size_t alignment = 0;
00204 };
00205
00209 struct data_provider {
00213     using alloc_func = std::function<data::ptr(size_t, size_t)>;
00214
00216     alloc_func on_alloc;
00217
00221     using free_func = std::function<void()>;
00222
00224     free_func on_free;
00225
00229     using realloc_func = std::function<data::ptr(data::ptr, size_t, size_t)>;
00230
00232     realloc_func on_realloc;
00233 };
00234
00238 struct c_data {
00240     using ref = c_data const&;
00241
00245     c_data() = default;
00246
00252     c_data(void const* addr,
00253           size_t length)
00254     : addr(data::as_ptr(addr)), size(length) {}
00255
00260     c_data(data::ref data)
00261     : c_data(data.addr, data.size) {}
00262
00264     data::c_ptr addr = nullptr;
00265
00267     size_t size = 0;
00268 };
00269
00273 struct u_data : data {
00275     using ref = u_data const&;
00276
00282     u_data(size_t length = 0,
00283           data::mode mode = data::mode::alloc) {
00284         if (length)
00285             set(length, mode);
00286     }
00287
00292     explicit u_data(data::ref data) {
00293         addr = data.addr;
00294         size = data.size;
00295         alignment = data.alignment;
00296     }
00297
00301     ~u_data() {
00302         deallocate();
00303     }
00304 };
00305
00311 inline size_t next_pow_2(size_t x) {

```

```

00312     x--;
00313     x |= x » 1;
00314     x |= x » 2;
00315     x |= x » 4;
00316     x |= x » 8;
00317     x |= x » 16;
00318     x++;
00319     return x;
00320 }
00321
00322 } // namespace lava

```

## 5.71 liblava/core/id.hpp File Reference

Object Identification.

```

#include "liblava/core/types.hpp"
#include <atomic>
#include <deque>
#include <memory>
#include <mutex>
#include <set>

```

### Classes

- struct [lava::id](#)  
*Identification.*
- struct [lava::ids](#)  
*Id factory.*
- struct [lava::id\\_listeners< T >](#)  
*Id listeners.*
- struct [lava::entity](#)  
*Entity.*
- struct [lava::id\\_registry< T, Meta >](#)  
*Id registry.*

### Typedefs

- using [lava::string\\_id\\_map](#) = std::map<[string](#), [id](#)>  
*Map of string ids.*

### Functions

- [id lava::to\\_id](#) (auto value)  
*Convert to id.*
- template<typename T >  
[id lava::add\\_id\\_map](#) (T const &object, std::map< [id](#), T > &map)  
*Add object to id map.*
- template<typename T >  
[bool lava::remove\\_id\\_map](#) ([id::ref](#) object\_id, std::map< [id](#), T > &map)  
*Remove object from id map.*



## Variables

- constexpr `id` const `lava::undef_id = id()`  
*Undefined id.*

### 5.71.1 Detailed Description

Object Identification.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.71.2 Function Documentation

#### 5.71.2.1 add\_id\_map()

```
template<typename T >
id lava::add_id_map (
    T const & object,
    std::map< id, T > & map) [inline]
```

Add object to id map.

#### Template Parameters

<i>T</i>	Type of object
----------	----------------

#### Parameters

<i>object</i>	Object to add
<i>map</i>	Target map

#### Returns

id Id of object in map

#### 5.71.2.2 remove\_id\_map()

```
template<typename T >
bool lava::remove_id_map (
    id::ref object_id,
    std::map< id, T > & map) [inline]
```

Remove object from id map.

### Template Parameters

<i>T</i>	Type of object
----------	----------------

### Parameters

<i>object</i> ↔ <i>_id</i>	Object to remove
<i>map</i>	Target map

### Returns

Removed object from map or object not found

#### 5.71.2.3 to\_id()

```
id lava::to_id (
    auto value) [inline]
```

Convert to id.

### Parameters

<i>value</i>	Source value
--------------	--------------

### Returns

id Converted value

## 5.72 id.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include <atomic>
00012 #include <deque>
00013 #include <memory>
00014 #include <mutex>
00015 #include <set>
00016
00017 namespace lava {
00018
00022 struct id {
00024     using ref = id const&;
00025
00027     using set = std::set<id>;
00028
00030     using set_ref = set const&;
00031
00033     using list = std::vector<id>;
00034
00036     using map = std::map<id, id>;
00037
00039     using index_map = std::map<id, index>;
00040
00042     using string_map = std::map<id, string>;
00043
```

```

00047     id() = default;
00048
00053     id(index value)
00054     : value(value) {}
00055
00057     index value = no_index;
00058
00063     bool valid() const {
00064         return value != no_index;
00065     }
00066
00071     string to_string() const {
00072         return std::to_string(value);
00073     }
00074
00078     void invalidate() {
00079         *this = {};
00080     }
00081
00085     auto operator<=>(id const&) const = default;
00086 };
00087
00089 using string_id_map = std::map<string, id>;
00090
00092 constexpr id const undef_id = id();
00093
00099 inline id to_id(auto value) {
00100     return {static_cast<index>(value)};
00101 }
00102
00106 struct ids {
00111     static ids& instance() {
00112         static ids ids;
00113         return ids;
00114     }
00115
00120     id next() {
00121         return {++m_next};
00122     }
00123
00124 private:
00126     std::atomic<index> m_next = {no_index};
00127 };
00128
00136 template <typename T>
00137 inline id add_id_map(T const& object,
00138                     std::map<id, T>& map) {
00139     auto next = ids::instance().next();
00140     map.emplace(next, std::move(object));
00141     return next;
00142 }
00143
00151 template <typename T>
00152 inline bool remove_id_map(id::ref object_id,
00153                          std::map<id, T>& map) {
00154     if (!map.count(object_id))
00155         return false;
00156
00157     map.erase(object_id);
00158
00159     return true;
00160 }
00161
00166 template <typename T>
00167 struct id_listeners {
00173     id add(typename T::func const& listener) {
00174         return add_id_map(listener, m_list);
00175     }
00176
00181     void remove(id& id) {
00182         if (remove_id_map(id, m_list))
00183             id.invalidate();
00184     }
00185
00190     typename T::listeners const& get_list() const {
00191         return m_list;
00192     }
00193
00194 private:
00196     typename T::listeners m_list = {};
00197 };
00198
00202 struct entity : no_copy_no_move, interface {
00206     entity()
00207     : m_id(ids::instance().next()) {}
00208
00213     id::ref get_id() const {

```

```

00214         return m_id;
00215     }
00216
00217 private:
00219     id m_id;
00220 };
00221
00227 template <typename T, typename Meta>
00228 struct id_registry {
00230     using s_ptr = std::shared_ptr<T>;
00231
00233     using s_map = std::map<id, s_ptr>;
00234
00236     using meta_map = std::map<id, Meta>;
00237
00243     id create(Meta info = {}) {
00244         auto object = std::make_shared<T>();
00245         add(object, info);
00246
00247         return object->get_id();
00248     }
00249
00255     void add(s_ptr object,
00256             Meta info = {}) {
00257         m_objects.emplace(object->get_id(), object);
00258         m_meta.emplace(object->get_id(), info);
00259     }
00260
00266     bool exists(id::ref object_id) const {
00267         return m_objects.count(object_id);
00268     }
00269
00275     s_ptr get(id::ref object_id) const {
00276         return m_objects.at(object_id);
00277     }
00278
00284     Meta const& get_meta(id::ref object_id) const {
00285         return m_meta.at(object_id);
00286     }
00287
00292     s_map const& get_all() const {
00293         return m_objects;
00294     }
00295
00300     meta_map const& get_all_meta() const {
00301         return m_meta;
00302     }
00303
00310     bool update(id::ref object_id,
00311               Meta const& meta) {
00312         if (!exists(object_id))
00313             return false;
00314
00315         m_meta.at(object_id) = meta;
00316         return true;
00317     }
00318
00323     void remove(id::ref object_id) {
00324         m_objects.erase(object_id);
00325         m_meta.erase(object_id);
00326     }
00327
00331     void clear() {
00332         m_objects.clear();
00333         m_meta.clear();
00334     }
00335
00336 private:
00338     s_map m_objects;
00339
00341     meta_map m_meta;
00342 };
00343
00344 } // namespace lava

```

## 5.73 liblava/core/misc.hpp File Reference

Miscellaneous helpers.

```

#include "liblava/core/types.hpp"
#include <algorithm>

```

```
#include <cstring>
#include <utility>
```

## Classes

- struct [lava::reversion\\_wrapper< T >](#)  
*Reversion Wrapper.*

## Functions

- bool [lava::exists](#) ([names\\_ref](#) list, [name](#) item)  
*Check if name exists in name list.*
- template<typename T >  
void [lava::remove](#) (std::vector< T > &list, T item)  
*Remove item from list.*
- template<typename T >  
bool [lava::contains](#) (std::vector< T > &list, T item)  
*Check if item is included in list.*
- template<typename T >  
void [lava::append](#) (std::vector< T > &list, std::vector< T > &items)  
*Append a list of items to another list.*
- void [lava::trim\\_start](#) (string &s)  
*Trim string only from start (in place)*
- void [lava::trim\\_end](#) (string &s)  
*Trim string only from end (in place)*
- void [lava::trim](#) (string &s)  
*Trim string from both ends (in place)*
- string [lava::trim\\_start\\_copy](#) (string s)  
*Trim string only from start (copying)*
- string [lava::trim\\_end\\_copy](#) (string s)  
*Trim string only from end (copying)*
- string [lava::trim\\_copy](#) (string s)  
*Trim string from both ends (copying)*
- string & [lava::remove\\_chars](#) (string &s, [string\\_ref](#) chars)  
*Remove chars in string.*
- string & [lava::remove\\_punctuation\\_marks](#) (string &s)  
*Remove punctuation marks in string.*
- string [lava::remove\\_chars\\_copy](#) (string s, [string\\_ref](#) chars)  
*Remove chars in string (copying)*
- string & [lava::remove\\_nondigit](#) (string &s)  
*Remove all non digit chars in string.*
- string [lava::remove\\_nondigit\\_copy](#) (string s)  
*Remove all non digit chars in string (copying)*
- string & [lava::remove\\_chars\\_if\\_not](#) (string &s, [string\\_ref](#) allowed)  
*Remove all chars in string which are not allowed.*
- string [lava::remove\\_chars\\_if\\_not\\_copy](#) (string s, [string\\_ref](#) allowed)  
*Remove all chars in string which are not allowed (copying)*
- template<typename T >  
auto [lava::begin](#) ([reversion\\_wrapper< T >](#) w)

- Begin the iterator.*
- `template<typename T >`  
`auto lava::end (reversion_wrapper< T > w)`
- End the iterator.*
- `template<typename T >`  
`reversion_wrapper< T > lava::reverse (T &&iterable)`
- Reverse iteration.*

## Variables

- `constexpr name lava::g_punctuation_marks_ = "\\\""`  
*Punctuation marks.*

## 5.73.1 Detailed Description

Miscellaneous helpers.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.73.2 Function Documentation

### 5.73.2.1 append()

```
template<typename T >
void lava::append (
    std::vector< T > & list,
    std::vector< T > & items) [inline]
```

Append a list of items to another list.

#### Template Parameters

<i>T</i>	Type of list
----------	--------------

#### Parameters

<i>list</i>	List of items
<i>items</i>	Items to append

### 5.73.2.2 begin()

```
template<typename T >
auto lava::begin (
    reversion_wrapper< T > w) [inline]
```

Begin the iterator.

## Template Parameters

<i>T</i>	Type of iterable
----------	------------------

## Parameters

<i>w</i>	Reversion wrapper
----------	-------------------

## Returns

auto Iterator

## 5.73.2.3 contains()

```
template<typename T >
bool lava::contains (
    std::vector< T > & list,
    T item) [inline]
```

Check if item is included in list.

## Template Parameters

<i>T</i>	Type of list
----------	--------------

## Parameters

<i>list</i>	List of items
<i>item</i>	Item to check

## Returns

Item exists or not found

## 5.73.2.4 end()

```
template<typename T >
auto lava::end (
    reversion\_wrapper< T > w) [inline]
```

End the iterator.

## Template Parameters

<i>T</i>	Type of iterable
----------	------------------

**Parameters**

<i>w</i>	Reversion wrapper
----------	-------------------

**Returns**

auto Iterator

**5.73.2.5 exists()**

```
bool lava::exists (  
    names_ref list,  
    name item) [inline]
```

Check if name exists in name list.

**Parameters**

<i>list</i>	List of names
<i>item</i>	Item to check

**Returns**

Item exists or not found

**5.73.2.6 remove()**

```
template<typename T >  
void lava::remove (  
    std::vector< T > & list,  
    T item) [inline]
```

Remove item from list.

**Template Parameters**

<i>T</i>	Type of list
----------	--------------

**Parameters**

<i>list</i>	List of items
<i>item</i>	Item to remove

**5.73.2.7 remove\_chars()**

```
string & lava::remove_chars (  
    string & s,  
    string_ref chars) [inline]
```

Remove chars in string.



## Parameters

<i>s</i>	Target string
<i>chars</i>	Chars to remove

## Returns

string& Cleared string

**5.73.2.8 remove\_chars\_copy()**

```
string lava::remove_chars_copy (  
    string s,  
    string_ref chars) [inline]
```

Remove chars in string (copying)

## Parameters

<i>s</i>	Target string
<i>chars</i>	Chars to remove

## Returns

string Cleared string

**5.73.2.9 remove\_chars\_if\_not()**

```
string & lava::remove_chars_if_not (  
    string & s,  
    string_ref allowed) [inline]
```

Remove all chars in string which are not allowed.

## Parameters

<i>s</i>	Target string
<i>allowed</i>	Allowed chars

## Returns

string& Cleared string

**5.73.2.10 remove\_chars\_if\_not\_copy()**

```
string lava::remove_chars_if_not_copy (  
    string s,  
    string_ref allowed) [inline]
```

Remove all chars in string which are not allowed (copying)

**Parameters**

<i>s</i>	Target string
<i>allowed</i>	Allowed chars

**Returns**

string Cleared string

**5.73.2.11 remove\_nondigit()**

```
string & lava::remove_nondigit (  
    string & s) [inline]
```

Remove all non digit chars in string.

**Parameters**

<i>s</i>	Target string
----------	---------------

**Returns**

string& Cleared string

**5.73.2.12 remove\_nondigit\_copy()**

```
string lava::remove_nondigit_copy (  
    string s) [inline]
```

Remove all non digit chars in string (copying)

**Parameters**

<i>s</i>	Target string
----------	---------------

**Returns**

string Cleared string

**5.73.2.13 remove\_punctuation\_marks()**

```
string & lava::remove_punctuation_marks (  
    string & s) [inline]
```

Remove punctuation marks in string.

## Parameters

<i>s</i>	Target string
----------	---------------

## Returns

string& Cleared string

**5.73.2.14 reverse()**

```
template<typename T >
reversion_wrapper< T > lava::reverse (
    T && iterable) [inline]
```

Reverse iteration.

## Template Parameters

<i>T</i>	Type of iterable
----------	------------------

## Parameters

<i>iterable</i>	Iterable
-----------------	----------

## Returns

reversion\_wrapper<T> Wrapper

**5.73.2.15 trim()**

```
void lava::trim (
    string & s) [inline]
```

Trim string from both ends (in place)

## Parameters

<i>s</i>	String to trim
----------	----------------

**5.73.2.16 trim\_copy()**

```
string lava::trim_copy (
    string s) [inline]
```

Trim string from both ends (copying)

**Parameters**

<code>s</code>	String to trim
----------------	----------------

**Returns**

string Trimmed string

**5.73.2.17 trim\_end()**

```
void lava::trim_end (  
    string & s) [inline]
```

Trim string only from end (in place)

**Parameters**

<code>s</code>	String to trim
----------------	----------------

**5.73.2.18 trim\_end\_copy()**

```
string lava::trim_end_copy (  
    string s) [inline]
```

Trim string only from end (copying)

**Parameters**

<code>s</code>	String to trim
----------------	----------------

**Returns**

string Trimmed string

**5.73.2.19 trim\_start()**

```
void lava::trim_start (  
    string & s) [inline]
```

Trim string only from start (in place)

**Parameters**

<code>s</code>	String to trim
----------------	----------------

**5.73.2.20 trim\_start\_copy()**

```
string lava::trim_start_copy (  
    string s) [inline]
```

Trim string only from start (copying)

## Parameters

s	String to trim
---	----------------

## Returns

string Trimmed string

## 5.74 misc.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include <algorithm>
00012 #include <cstring>
00013 #include <utility>
00014
00015 namespace lava {
00016
00023 inline bool exists(names_ref list, name item) {
00024     auto itr = std::find_if(list.begin(), list.end(),
00025                             [&](name entry) { return strcmp(entry, item) == 0; });
00026     return itr != list.end();
00027 }
00028
00035 template <typename T>
00036 inline void remove(std::vector<T>& list, T item) {
00037     list.erase(std::remove(list.begin(), list.end(), item), list.end());
00038 }
00039
00047 template <typename T>
00048 inline bool contains(std::vector<T>& list, T item) {
00049     return std::find(list.begin(), list.end(), item) != list.end();
00050 }
00051
00058 template <typename T>
00059 inline void append(std::vector<T>& list, std::vector<T>& items) {
00060     list.insert(list.end(), items.begin(), items.end());
00061 }
00062
00067 inline void trim_start(string& s) {
00068     s.erase(s.begin(), std::find_if(s.begin(), s.end(), [](uchar ch) {
00069         return !std::isspace(ch);
00070     }));
00071 }
00072
00077 inline void trim_end(string& s) {
00078     s.erase(std::find_if(s.rbegin(), s.rend(), [](uchar ch) {
00079         return !std::isspace(ch);
00080     }).base(),
00081             s.end());
00082 }
00083
00088 inline void trim(string& s) {
00089     trim_start(s);
00090     trim_end(s);
00091 }
00092
00098 inline string trim_start_copy(string s) {
00099     trim_start(s);
00100     return s;
00101 }
00102
00108 inline string trim_end_copy(string s) {
00109     trim_end(s);
00110     return s;
00111 }
00112
00118 inline string trim_copy(string s) {
00119     trim(s);
00120     return s;
00121 }
00122
00129 inline string& remove_chars(string& s, string_ref chars) {

```

```

00130     s.erase(std::remove_if(s.begin(), s.end(), [&chars](name_ref c) {
00131         return chars.find(c) != string::npos;
00132     })),
00133     s.end());
00134     return s;
00135 }
00136
00137 constexpr name g_punctuation_marks_ = "\\'\";
00138
00139 inline string& remove_punctuation_marks(string& s) {
00140     return remove_chars(s, g_punctuation_marks_);
00141 }
00142
00143 inline string remove_chars_copy(string s, string_ref chars) {
00144     return remove_chars(s, chars);
00145 }
00146
00147 inline string& remove_nondigit(string& s) {
00148     s.erase(std::remove_if(s.begin(), s.end(), [](name_ref c) {
00149         return !isdigit(c);
00150     })),
00151     s.end());
00152     return s;
00153 }
00154
00155 inline string remove_nondigit_copy(string s) {
00156     return remove_nondigit(s);
00157 }
00158
00159 inline string& remove_chars_if_not(string& s, string_ref allowed) {
00160     s.erase(std::remove_if(s.begin(), s.end(), [&allowed](name_ref c) {
00161         return allowed.find(c) == string::npos;
00162     })),
00163     s.end());
00164     return s;
00165 }
00166
00167 inline string remove_chars_if_not_copy(string s, string_ref allowed) {
00168     return remove_chars_if_not(s, allowed);
00169 }
00170
00171 template <typename T>
00172 struct reversion_wrapper {
00173     T& iterable;
00174 };
00175
00176 template <typename T>
00177 inline auto begin(reversion_wrapper<T> w) {
00178     return std::rbegin(w.iterable);
00179 }
00180
00181 template <typename T>
00182 inline auto end(reversion_wrapper<T> w) {
00183     return std::rend(w.iterable);
00184 }
00185
00186 template <typename T>
00187 inline reversion_wrapper<T> reverse(T&& iterable) {
00188     return {iterable};
00189 }
00190
00191 } // namespace lava

```

## 5.75 liblava/core/time.hpp File Reference

Run time.

```

#include "liblava/core/types.hpp"
#include <chrono>
#include <iomanip>
#include <sstream>

```

### Classes

- struct [lava::timer](#)

*Timer.*

- struct [lava::run\\_time](#)

*Run time.*

## Typedefs

- using **lava::seconds** = std::chrono::seconds  
*Seconds.*
- using **lava::milliseconds** = std::chrono::milliseconds  
*Milliseconds.*
- using **lava::ms** = [milliseconds](#)  
*Milliseconds.*
- using **lava::microseconds** = std::chrono::microseconds  
*Microseconds.*
- using **lava::us** = [microseconds](#)  
*Microseconds.*
- using **lava::clock** = std::chrono::high\_resolution\_clock  
*Clock.*
- using **lava::time\_point** = clock::time\_point  
*Time point.*
- using **lava::duration** = clock::duration  
*Duration.*

## Functions

- [delta lava::to\\_delta](#) ([milliseconds ms](#))  
*Convert milliseconds to delta.*
- [delta lava::to\\_dt](#) ([milliseconds ms](#))
- [real lava::to\\_sec](#) ([milliseconds ms](#))  
*Convert milliseconds to seconds.*
- [i32 lava::to\\_sec\\_fix](#) ([milliseconds ms](#))  
*Convert milliseconds to fixed seconds.*
- [ms lava::to\\_ms](#) ([delta dt](#))  
*Convert delta to milliseconds.*
- [ms lava::to\\_ms](#) ([real sec](#))  
*Convert seconds to milliseconds.*
- template<typename CLOCK\_TYPE = std::chrono::system\_clock>  
[string lava::timestamp](#) (const typename CLOCK\_TYPE::time\_point &[time\\_point](#), [string\\_ref](#) format="%Y-%m-%d %H-%M-%S")  
*Convert timestamp to string.*
- [string lava::get\\_current\\_time](#) ()  
*Get the current time as string.*
- [ms lava::get\\_current\\_timestamp\\_ms](#) ()  
*Get the current timestamp in milliseconds.*
- [us lava::get\\_current\\_timestamp\\_us](#) ()  
*Get the current timestamp in microseconds.*
- [ui64 lava::get\\_current\\_timestamp](#) ()  
*Get the current timestamp in milliseconds (uint)*

## Variables

- constexpr `seconds` const `lava::one_second` = `seconds`(1)  
*One second.*
- constexpr `ms` const `lava::one_ms` = `ms`(1)  
*One millisecond.*
- constexpr `us` const `lava::one_us` = `us`(1)  
*One microsecond.*

### 5.75.1 Detailed Description

Run time.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.75.2 Function Documentation

#### 5.75.2.1 `get_current_time()`

```
string lava::get_current_time () [inline]
```

Get the current time as string.

#### Returns

string Time and date representation

#### 5.75.2.2 `get_current_timestamp()`

```
ui64 lava::get_current_timestamp () [inline]
```

Get the current timestamp in milliseconds (uint)

#### Returns

ui64 Timestamp in ms (uint)



### 5.75.2.3 get\_current\_timestamp\_ms()

```
ms lava::get_current_timestamp_ms () [inline]
```

Get the current timestamp in milliseconds.

#### Returns

ms Timestamp in ms

### 5.75.2.4 get\_current\_timestamp\_us()

```
us lava::get_current_timestamp_us () [inline]
```

Get the current timestamp in microseconds.

#### Returns

us Timestamp in us

### 5.75.2.5 timestamp()

```
template<typename CLOCK_TYPE = std::chrono::system_clock>
string lava::timestamp (
    const typename CLOCK_TYPE::time_point & time_point,
    string_ref format = "%Y-%m-%d %H-%M-%S") [inline]
```

Convert timestamp to string.

#### Template Parameters

<i>CLOCK_TYPE</i>	Clock type
-------------------	------------

#### Parameters

<i>time_point</i>	Time point
<i>format</i>	String format

#### Returns

string Converted string

### 5.75.2.6 to\_delta()

```
delta lava::to_delta (
    milliseconds ms) [inline]
```

Convert milliseconds to delta.

**Parameters**

<i>ms</i>	Milliseconds to convert
-----------	-------------------------

**Returns**

delta Converted delta

**5.75.2.7 to\_dt()**

```
delta lava::to_dt (
    milliseconds ms) [inline]
```

**See also**

to\_delta()

**5.75.2.8 to\_ms() [1/2]**

```
ms lava::to_ms (
    delta dt) [inline]
```

Convert delta to milliseconds.

**Parameters**

<i>dt</i>	Delta to convert
-----------	------------------

**Returns**

ms Converted milliseconds

**5.75.2.9 to\_ms() [2/2]**

```
ms lava::to_ms (
    real sec) [inline]
```

Convert seconds to milliseconds.

**Parameters**

<i>sec</i>	Seconds to convert
------------	--------------------

**Returns**

ms Converted milliseconds

**5.75.2.10 to\_sec()**

```
real lava::to_sec (
    milliseconds ms) [inline]
```

Convert milliseconds to seconds.

## Parameters

<i>ms</i>	Milliseconds to convert
-----------	-------------------------

## Returns

real Converted seconds

## 5.75.2.11 to\_sec\_fix()

```
i32 lava::to_sec_fix (
    milliseconds ms) [inline]
```

Convert milliseconds to fixed seconds.

## Parameters

<i>ms</i>	Milliseconds to convert
-----------	-------------------------

## Returns

i32 Converted fixed seconds

## 5.76 time.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include <chrono>
00012 #include <iomanip>
00013 #include <sstream>
00014
00015 namespace lava {
00016
00018 using namespace std::chrono_literals;
00019
00021 using seconds = std::chrono::seconds;
00022
00024 using milliseconds = std::chrono::milliseconds;
00025
00027 using ms = milliseconds;
00028
00030 using microseconds = std::chrono::microseconds;
00031
00033 using us = microseconds;
00034
00036 constexpr seconds const one_second = seconds(1);
00037
00039 constexpr ms const one_ms = ms(1);
00040
00042 constexpr us const one_us = us(1);
00043
00045 using clock = std::chrono::high_resolution_clock;
00046
00048 using time_point = clock::time_point;
00049
00051 using duration = clock::duration;
00052
00058 inline delta to_delta(milliseconds ms) {
00059     return ms.count() / 1000.f;
00060 }
00061
```

```

00065 inline delta to_dt(milliseconds ms) {
00066     return to_delta(ms);
00067 }
00068
00074 inline real to_sec(milliseconds ms) {
00075     return ms.count() / 1000.;
00076 }
00077
00083 inline i32 to_sec_fix(milliseconds ms) {
00084     return to_i32(to_sec(ms));
00085 }
00086
00092 inline ms to_ms(delta dt) {
00093     return ms(to_i32(dt * 1000.f));
00094 }
00095
00101 inline ms to_ms(real sec) {
00102     return ms(to_i32(sec * 1000.));
00103 }
00104
00108 struct timer {
00112     timer()
00113     : m_start_time(clock::now()) {}
00114
00118     void reset() {
00119         m_start_time = clock::now();
00120     }
00121
00126     ms elapsed() const {
00127         return std::chrono::duration_cast<ms>(clock::now()
00128                                             - m_start_time);
00129     }
00130
00131 private:
00133     time_point m_start_time;
00134 };
00135
00139 struct run_time {
00141     ms current{0};
00142
00144     ms clock{16};
00145
00147     ms system{0};
00148
00150     ms delta{0};
00151
00153     ms fix_delta{0};
00154
00156     r32 speed = 1.f;
00157
00159     bool paused = false;
00160 };
00161
00162 #ifdef _WIN32
00163     #pragma warning(push)
00164     #pragma warning(disable : 4996) //_CRT_SECURE_NO_WARNINGS
00165 #endif
00166
00174 template <typename CLOCK_TYPE = std::chrono::system_clock>
00175 inline string timestamp(const typename CLOCK_TYPE::time_point& time_point,
00176                        string_ref format = "%Y-%m-%d %H-%M-%S") {
00177     auto ms = std::chrono::duration_cast<milliseconds>(
00178         time_point.time_since_epoch())
00179         % 1000;
00180
00181     auto const t = CLOCK_TYPE::to_time_t(time_point);
00182     auto const tm = *std::localtime(std::addressof(t));
00183
00184     std::ostringstream stm;
00185     stm << std::put_time(std::addressof(tm), str(format))
00186         << "." << std::setfill('0') << std::setw(3) << ms.count();
00187     return stm.str();
00188 }
00189
00194 inline string get_current_time() {
00195     auto now = std::chrono::system_clock::now();
00196     return timestamp(now);
00197 }
00198
00199 #ifdef _WIN32
00200     #pragma warning(pop)
00201 #endif
00202
00207 inline ms get_current_timestamp_ms() {
00208     return std::chrono::duration_cast<ms>(
00209         clock::now().time_since_epoch());
00210 }

```

```

00211
00216 inline us get_current_timestamp_us() {
00217     return std::chrono::duration_cast<us>(
00218         clock::now().time_since_epoch());
00219 }
00220
00225 inline ui64 get_current_timestamp() {
00226     return get_current_timestamp_ms().count();
00227 }
00228
00229 } // namespace lava

```

## 5.77 liblava/core/types.hpp File Reference

Basic types.

```

#include "liblava/core/def.hpp"
#include <cstdint>
#include <functional>
#include <map>
#include <string>
#include <string_view>
#include <vector>

```

### Classes

- struct [lava::no\\_copy\\_no\\_move](#)  
*No copy and no move object.*
- struct [lava::interface](#)  
*Interface.*
- struct [lava::pair\\_hash](#)  
*Pair hash.*

### Macros

- #define [ENUM\\_FLAG\\_OPERATORS](#)(T)  
*Enum flag operators.*

### Typedefs

- using [lava::int8](#) = std::int8\_t  
*8 bit integer*
- using [lava::i8](#) = [int8](#)
- using [lava::uint8](#) = std::uint8\_t  
*8 bit unsigned integer*
- using [lava::ui8](#) = [uint8](#)
- using [lava::int16](#) = std::int16\_t  
*16 bit integer*
- using [lava::i16](#) = [int16](#)
- using [lava::uint16](#) = std::uint16\_t  
*16 bit unsigned integer*
- using [lava::ui16](#) = [uint16](#)

- using **lava::int32** = std::int32\_t  
*32 bit integer*
- using **lava::i32** = int32
- using **lava::uint32** = std::uint32\_t  
*32 bit unsigned integer*
- using **lava::ui32** = uint32
- using **lava::int64** = std::int64\_t  
*64 bit integer*
- using **lava::i64** = int64
- using **lava::uint64** = std::uint64\_t  
*64 bit unsigned integer*
- using **lava::ui64** = uint64
- using **lava::char8** = std::int\_fast8\_t  
*8 bit char*
- using **lava::c8** = char8
- using **lava::uchar8** = std::uint\_fast8\_t  
*8 bit unsigned char*
- using **lava::uc8** = uchar8
- using **lava::char16** = int16  
*16 bit char*
- using **lava::c16** = char16
- using **lava::uchar16** = uint16  
*16 bit unsigned char*
- using **lava::uc16** = uchar16
- using **lava::char32** = int32  
*32 bit char*
- using **lava::c32** = char32
- using **lava::uchar32** = uint32  
*32 bit unsigned char*
- using **lava::uc32** = uchar32
- using **lava::size\_t** = std::size\_t  
*Size.*
- using **lava::uchar** = unsigned char  
*Unsigned char.*
- using **lava::r32** = float  
*Single-precision floating-point.*
- using **lava::r64** = double  
*Double-precision floating-point.*
- using **lava::real** = r64  
*Real number.*
- using **lava::delta** = r32  
*Delta.*
- using **lava::void\_ptr** = void\*  
*Void pointer.*
- using **lava::void\_c\_ptr** = void const\*  
*Const void pointer.*
- using **lava::flag** = ui32  
*Flag.*
- using **lava::index** = ui32  
*Index.*
- using **lava::index\_list** = std::vector<index>

- List of indices.*
- using **lava::index\_map** = std::map<index, index>
- Map of indices.*
- using **lava::string** = std::string
- String.*
- using **lava::string\_ref** = string const&
- Reference to string.*
- using **lava::string\_list** = std::vector<string>
- List of strings.*
- using **lava::string\_list\_ref** = string\_list const&
- Reference to list of strings.*
- using **lava::string\_view** = std::string\_view
- String view.*
- using **lava::string\_map** = std::map<string, string>
- Map of strings.*
- using **lava::string\_map\_ref** = string\_map const&
- Reference to map of strings.*
- using **lava::name** = char const\*
- Name.*
- using **lava::name\_ref** = char const&
- Reference to name.*
- using **lava::names** = std::vector<name>
- List of names.*
- using **lava::names\_ref** = names const&
- Reference to list of names.*

## Functions

- **name lava::str** (string\_ref value)
- Get c-string representation of string.*
- **r32 lava::to\_r32** (auto value)
- Convert to r32.*
- **r64 lava::to\_r64** (auto value)
- Convert to r64.*
- **i32 lava::to\_i32** (auto value)
- Convert to i32.*
- **i64 lava::to\_i64** (auto value)
- Convert to i64.*
- **ui32 lava::to\_ui32** (auto value)
- Convert to ui32.*
- **ui64 lava::to\_ui64** (auto value)
- Convert to ui64.*
- **size\_t lava::to\_size\_t** (auto value)
- Convert to size\_t.*
- **index lava::to\_index** (auto value)
- Convert to index.*
- **char const \* lava::to\_char** (auto value)
- Convert to pointer const char.*
- **template<typename T >**  
void **lava::hash\_combine** (size\_t &seed, T const &val)

*Combine hash seed with value - from boost (functional/hash)*

- `template<typename T >`  
`void lava::hash\_value (size\_t &seed, T const &val)`  
*Auxiliary generic functions to create a hash value using a seed.*
- `template<typename T, typename... Types>`  
`void lava::hash\_value (size\_t &seed, T const &val, Types const &... args)`
- `template<typename... Types>`  
`size\_t lava::hash\_value (Types const &... args)`

## Variables

- `constexpr i32 const lava::undef = -1`  
*Undefined value.*
- `constexpr index const lava::no\_index = 0xffffffff`  
*No index.*
- `constexpr name lava::\_lava = "lava"`  
*lava*
- `constexpr name lava::\_liblava = "liblava"`  
*liblava*
- `constexpr name lava::\_default = "default"`  
*default*

## 5.77.1 Detailed Description

Basic types.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.77.2 Macro Definition Documentation

### 5.77.2.1 ENUM\_FLAG\_OPERATORS

```
#define ENUM_FLAG_OPERATORS(  
    T)
```

#### Value:

```
inline T operator~(T a) { \
    return static_cast<T>(~static_cast<std::underlying_type<T>::type>(a)); \
} \
inline T operator|(T a, T b) { \
    return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) | \
        static_cast<std::underlying_type<T>::type>(b)); \
} \
inline T operator&(T a, T b) { \
    return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) & \
        static_cast<std::underlying_type<T>::type>(b)); \
} \
inline T operator^(T a, T b) { \
```



```

        return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) ^
        static_cast<std::underlying_type<T>::type>(b)); \
    } \
    inline T& operator|=(T& a, T b) { \
        return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) |=
        static_cast<std::underlying_type<T>::type>(b)); \
    } \
    inline T& operator&=(T& a, T b) { \
        return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) &=
        static_cast<std::underlying_type<T>::type>(b)); \
    } \
    inline T& operator^=(T& a, T b) { \
        return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) ^=
        static_cast<std::underlying_type<T>::type>(b)); \
    }

```

Enum flag operators.

## 5.77.3 Typedef Documentation

### 5.77.3.1 c16

```
using lava::c16 = char16
```

See also

char16

### 5.77.3.2 c32

```
using lava::c32 = char32
```

See also

char32

### 5.77.3.3 c8

```
using lava::c8 = char8
```

See also

char8

### 5.77.3.4 i16

```
using lava::i16 = int16
```

See also

int16

#### 5.77.3.5 i32

```
using lava::i32 = int32
```

See also

int32

#### 5.77.3.6 i64

```
using lava::i64 = int64
```

See also

int64

#### 5.77.3.7 i8

```
using lava::i8 = int8
```

See also

int8

#### 5.77.3.8 uc16

```
using lava::uc16 = uchar16
```

See also

uchar16

#### 5.77.3.9 uc32

```
using lava::uc32 = uchar32
```

See also

uchar32

#### 5.77.3.10 uc8

```
using lava::uc8 = uchar8
```

See also

uchar8

### 5.77.3.11 ui16

```
using lava::ui16 = uint16
```

See also

uint16

### 5.77.3.12 ui32

```
using lava::ui32 = uint32
```

See also

uint32

### 5.77.3.13 ui64

```
using lava::ui64 = uint64
```

See also

uint64

### 5.77.3.14 ui8

```
using lava::ui8 = uint8
```

See also

uint8

## 5.77.4 Function Documentation

### 5.77.4.1 hash\_combine()

```
template<typename T >
void lava::hash_combine (
    size_t & seed,
    T const & val) [inline]
```

Combine hash seed with value - from boost (functional/hash)

See also

[https://www.boost.org/doc/libs/1\\_77\\_0/doc/html/hash/combine.html](https://www.boost.org/doc/libs/1_77_0/doc/html/hash/combine.html)

## Template Parameters

<i>T</i>	Type of value
----------	---------------

## Parameters

<i>seed</i>	Seed to combine
<i>val</i>	Value to combine

5.77.4.2 `hash_value()` [1/3]

```
template<typename T >
void lava::hash_value (
    size_t & seed,
    T const & val) [inline]
```

Auxiliary generic functions to create a hash value using a seed.

## Template Parameters

<i>T</i>	Type of value
----------	---------------

## Parameters

<i>seed</i>	Seed for hash
<i>val</i>	Hash value

5.77.4.3 `hash_value()` [2/3]

```
template<typename T , typename... Types>
void lava::hash_value (
    size_t & seed,
    T const & val,
    Types const &... args) [inline]
```

## See also

`hash_value<T>()`

5.77.4.4 `hash_value()` [3/3]

```
template<typename... Types>
size_t lava::hash_value (
    Types const &... args) [inline]
```

## See also

`hash_value<T>()`

5.77.4.5 `str()`

```
name lava::str (
    string_ref value) [inline]
```

Get c-string representation of string.

**Parameters**

<i>value</i>	Source string
--------------	---------------

**Returns**

name C-string representation

**5.77.4.6 to\_char()**

```
char const * lava::to_char (  
    auto value) [inline]
```

Convert to pointer const char.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

char const\* Converted value

**5.77.4.7 to\_i32()**

```
i32 lava::to_i32 (  
    auto value) [inline]
```

Convert to i32.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

i32 Converted value

**5.77.4.8 to\_i64()**

```
i64 lava::to_i64 (  
    auto value) [inline]
```

Convert to i64.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

i64 Converted value

**5.77.4.9 to\_index()**

```
index lava::to_index (  
    auto value) [inline]
```

Convert to index.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

index Converted value

**5.77.4.10 to\_r32()**

```
r32 lava::to_r32 (  
    auto value) [inline]
```

Convert to r32.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

r32 Converted value

**5.77.4.11 to\_r64()**

```
r64 lava::to_r64 (  
    auto value) [inline]
```

Convert to r64.

## Parameters

<i>value</i>	Source value
--------------	--------------

## Returns

r64 Converted value

**5.77.4.12 to\_size\_t()**

```
size_t lava::to_size_t (  
    auto value) [inline]
```

Convert to size\_t.

## Parameters

<i>value</i>	Source value
--------------	--------------

## Returns

size\_t Converted value

**5.77.4.13 to\_ui32()**

```
ui32 lava::to_ui32 (  
    auto value) [inline]
```

Convert to ui32.

## Parameters

<i>value</i>	Source value
--------------	--------------

## Returns

ui32 Converted value

**5.77.4.14 to\_ui64()**

```
ui64 lava::to_ui64 (  
    auto value) [inline]
```

Convert to ui64.

**Parameters**

<i>value</i>	Source value
--------------	--------------

**Returns**

ui64 Converted value

**5.78 types.hpp**

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/def.hpp"
00011 #include <cstdint>
00012 #include <functional>
00013 #include <map>
00014 #include <string>
00015 #include <string_view>
00016 #include <vector>
00017
00018 namespace lava {
00019
00021 using int8 = std::int8_t;
00022
00024 using i8 = int8;
00025
00027 using uint8 = std::uint8_t;
00028
00030 using ui8 = uint8;
00031
00033 using int16 = std::int16_t;
00034
00036 using i16 = int16;
00037
00039 using uint16 = std::uint16_t;
00040
00042 using ui16 = uint16;
00043
00045 using int32 = std::int32_t;
00046
00048 using i32 = int32;
00049
00051 using uint32 = std::uint32_t;
00052
00054 using ui32 = uint32;
00055
00057 using int64 = std::int64_t;
00058
00060 using i64 = int64;
00061
00063 using uint64 = std::uint64_t;
00064
00066 using ui64 = uint64;
00067
00069 using char8 = std::int_fast8_t;
00070
00072 using c8 = char8;
00073
00075 using uchar8 = std::uint_fast8_t;
00076
00078 using uc8 = uchar8;
00079
00081 using char16 = int16;
00082
00084 using c16 = char16;
00085
00087 using uchar16 = uint16;
00088
00090 using uc16 = uchar16;
00091
00093 using char32 = int32;
00094
00096 using c32 = char32;
00097
00099 using uchar32 = uint32;

```



```

00100
00102 using uc32 = uchar32;
00103
00105 using size_t = std::size_t;
00106
00108 using uchar = unsigned char;
00109
00111 using r32 = float;
00112
00114 using r64 = double;
00115
00117 using real = r64;
00118
00120 using delta = r32;
00121
00123 using void_ptr = void*;
00124
00126 using void_c_ptr = void const*;
00127
00129 constexpr i32 const undef = -1;
00130
00132 using flag = ui32;
00133
00135 using index = ui32;
00136
00138 constexpr index const no_index = 0xffffffff;
00139
00141 using index_list = std::vector<index>;
00142
00144 using index_map = std::map<index, index>;
00145
00147 using string = std::string;
00148
00150 using string_ref = string const&;
00151
00153 using string_list = std::vector<string>;
00154
00156 using string_list_ref = string_list const&;
00157
00159 using string_view = std::string_view;
00160
00162 using string_map = std::map<string, string>;
00163
00165 using string_map_ref = string_map const&;
00166
00168 using name = char const*;
00169
00171 using name_ref = char const&;
00172
00174 using names = std::vector<name>;
00175
00177 using names_ref = names const&;
00178
00180 constexpr name _lava_ = "lava";
00181
00183 constexpr name _liblava_ = "liblava";
00184
00186 constexpr name _default_ = "default";
00187
00193 inline name str(string_ref value) {
00194     return value.c_str();
00195 }
00196
00202 inline r32 to_r32(auto value) {
00203     return static_cast<r32>(value);
00204 }
00205
00211 inline r64 to_r64(auto value) {
00212     return static_cast<r64>(value);
00213 }
00214
00220 inline i32 to_i32(auto value) {
00221     return static_cast<i32>(value);
00222 }
00223
00229 inline i64 to_i64(auto value) {
00230     return static_cast<i64>(value);
00231 }
00232
00238 inline ui32 to_ui32(auto value) {
00239     return static_cast<ui32>(value);
00240 }
00241
00247 inline ui64 to_ui64(auto value) {
00248     return static_cast<ui64>(value);
00249 }
00250

```

```

00256 inline size_t to_size_t(auto value) {
00257     return static_cast<size_t>(value);
00258 }
00259
00260 inline index to_index(auto value) {
00261     return static_cast<index>(value);
00262 }
00263
00264 inline char const* to_char(auto value) {
00265     return (char const*)value;
00266 }
00267
00268
00269 struct no_copy_no_move {
00270     no_copy_no_move() = default;
00271
00272     no_copy_no_move(no_copy_no_move const&) = delete;
00273
00274     void operator=(no_copy_no_move const&) = delete;
00275 };
00276
00277 struct interface {
00278     virtual ~interface() = default;
00279 };
00280
00281 template <typename T>
00282 inline void hash_combine(size_t& seed,
00283                          T const& val) {
00284     seed ^= std::hash<T>()(val) + 0x9e3779b9
00285            + (seed « 6) + (seed » 2);
00286 }
00287
00288 template <typename T>
00289 inline void hash_value(size_t& seed,
00290                        T const& val) {
00291     hash_combine(seed, val);
00292 }
00293
00294 template <typename T, typename... Types>
00295 inline void hash_value(size_t& seed,
00296                        T const& val,
00297                        Types const&... args) {
00298     hash_combine(seed, val);
00299     hash_value(seed, args...);
00300 }
00301
00302 template <typename... Types>
00303 inline size_t hash_value(Types const&... args) {
00304     size_t seed = 0;
00305     hash_value(seed, args...);
00306     return seed;
00307 }
00308
00309 struct pair_hash {
00310     template <class T1, class T2>
00311     size_t operator()(std::pair<T1, T2> const& p) const {
00312         return hash_value(p.first, p.second);
00313     }
00314 };
00315
00316 #define ENUM_FLAG_OPERATORS(T) \
00317     inline T operator~(T a) { \
00318         return static_cast<T>(~static_cast<std::underlying_type<T>::type>(a)); \
00319     } \
00320     inline T operator|(T a, T b) { \
00321         return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) | \
00322                                static_cast<std::underlying_type<T>::type>(b)); \
00323     } \
00324     inline T operator&(T a, T b) { \
00325         return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) & \
00326                                static_cast<std::underlying_type<T>::type>(b)); \
00327     } \
00328     inline T operator^(T a, T b) { \
00329         return static_cast<T>(static_cast<std::underlying_type<T>::type>(a) ^ \
00330                                static_cast<std::underlying_type<T>::type>(b)); \
00331     } \
00332     inline T& operator|=(T& a, T b) { \
00333         return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) |= \
00334                                     static_cast<std::underlying_type<T>::type>(b)); \
00335     } \
00336     inline T& operator&=(T& a, T b) { \
00337         return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) &= \
00338                                     static_cast<std::underlying_type<T>::type>(b)); \
00339     } \
00340     inline T& operator^=(T& a, T b) { \
00341         return reinterpret_cast<T&>(reinterpret_cast<std::underlying_type<T>::type&>(a) ^= \
00342                                     static_cast<std::underlying_type<T>::type>(b)); \
00343     }

```

```
00395
00396 } // namespace lava
```

## 5.79 liblava/core/version.hpp File Reference

Version information.

```
#include "liblava/core/types.hpp"
```

### Classes

- struct [lava::semantic\\_version](#)  
*Semantic version.*
- struct [lava::version](#)  
*Version.*

### Typedefs

- using **lava::sem\_version** = [semantic\\_version](#)  
*Semantic version.*

### Enumerations

- enum class [lava::version\\_stage](#) : index {  
**preview** , **alpha** , **beta** , **rc** ,  
**release** , **rolling** }  
*Version stages.*

### Functions

- [sem\\_version](#) [lava::to\\_version](#) (string\_ref str)  
*Convert string to semantic version.*

### Variables

- constexpr **name** [lava::g\\_build\\_date](#) = LAVA\_BUILD\_DATE  
*Build date.*
- constexpr **name** [lava::g\\_build\\_time](#) = LAVA\_BUILD\_TIME  
*Build time.*

### 5.79.1 Detailed Description

Version information.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.79.2 Function Documentation

#### 5.79.2.1 to\_version()

```
sem_version lava::to_version (
    string_ref str) [inline]
```

Convert string to semantic version.

#### Parameters

<i>str</i>	String to convert
------------	-------------------

#### Returns

sem\_version Semantic version

## 5.80 version.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011
00012 namespace lava {
00013
00017 struct semantic_version {
00019     ui32 major = LAVA_VERSION_MAJOR;
00020
00022     ui32 minor = LAVA_VERSION_MINOR;
00023
00025     ui32 patch = LAVA_VERSION_PATCH;
00026
00030     auto operator<=>(semantic_version const&) const = default;
00031 };
00032
00034 using sem_version = semantic_version;
00035
00041 inline sem_version to_version(string_ref str) {
00042     sem_version result{0, 0, 0};
00043     #ifdef _WIN32
00044         sscanf_s
00045     #else
00046         sscanf
00047     #endif
00048         (str.c_str(), "%d.%d.%d", &result.major, &result.minor, &result.patch);
00049     return result;
00050 }
```

```

00051
00055 enum class version_stage : index {
00056     preview,
00057     alpha,
00058     beta,
00059     rc,
00060     release,
00061     rolling
00062 };
00063
00067 struct version {
00069     ui32 year = 2024;
00070
00072     ui32 release = 0;
00073
00075     version_stage stage = version_stage::rolling;
00076
00078     ui32 rev = 0;
00079 };
00080
00082 constexpr name g_build_date = LAVA_BUILD_DATE;
00083
00085 constexpr name g_build_time = LAVA_BUILD_TIME;
00086
00087 } // namespace lava

```

## 5.81 liblava/engine.hpp File Reference

Engine module.

```

#include "liblava/engine/engine.hpp"
#include "liblava/engine/producer.hpp"
#include "liblava/engine/props.hpp"

```

### 5.81.1 Detailed Description

Engine module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.82 engine.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/engine/engine.hpp"
00011 #include "liblava/engine/producer.hpp"
00012 #include "liblava/engine/props.hpp"

```

## 5.83 liblava/engine/engine.hpp File Reference

Engine.

```
#include "liblava/app/app.hpp"
#include "liblava/engine/producer.hpp"
#include "liblava/engine/props.hpp"
```

### Classes

- struct [lava::engine](#)

*Engine.*

### Variables

- constexpr [name](#) [lava::\\_props\\_](#) = "props"

*config*

### 5.83.1 Detailed Description

Engine.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.84 engine.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/app/app.hpp"
00011 #include "liblava/engine/producer.hpp"
00012 #include "liblava/engine/props.hpp"
00013
00014 namespace lava {
00015
00017 constexpr name _props_ = "props";
00018
00022 struct engine : app {
00023     // Pointer to engine
00024     using ptr = engine*;
00025
00027     using app::app;
00028
00033     bool setup() override;
00034
00036     lava::props props;
00037
00039     lava::producer producer;
00040
```

```

00042     using hud_menu_func = std::function<void()>;
00043
00044     hud_menu_func on_menu;
00045
00046     bool hud_active = false;
00047
00048 private:
00049     void handle_config();
00050
00051     void hud_menu();
00052
00053     id m_menu_layer;
00054
00055     #if LAVA_DEBUG
00056         id m_demo_layer;
00057     #endif
00058
00059     json_file::callback m_config_callback;
00060 };
00061
00062 } // namespace lava

```

## 5.85 liblava/engine/producer.hpp File Reference

Producer.

```

#include "liblava/fwd.hpp"
#include "liblava/resource.hpp"

```

### Classes

- struct [lava::producer](#)  
*Producer.*

### Variables

- constexpr [name](#) [lava::\\_shader\\_path\\_](#) = "shader/"  
*shader folder*
- constexpr [name](#) [lava::\\_temp\\_path\\_](#) = "temp/"  
*temp folder*
- constexpr [name](#) [lava::\\_hash\\_json\\_](#) = "hash.json"  
*hash file*

### 5.85.1 Detailed Description

Producer.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.86 producer.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/fwd.hpp"
00011 #include "liblava/resource.hpp"
00012
00013 namespace lava {
00014
00016 constexpr name _shader_path_ = "shader/";
00017
00019 constexpr name _temp_path_ = "temp/";
00020
00022 constexpr name _hash_json_ = "hash.json";
00023
00027 struct producer {
00029     using ptr = producer*;
00030
00032     engine* app = nullptr;
00033
00039     mesh::s_ptr create_mesh(mesh_type mesh_type);
00040
00046     mesh::s_ptr get_mesh(string_ref name);
00047
00053     bool add_mesh(mesh::s_ptr mesh);
00054
00060     texture::s_ptr create_texture(uv2 size);
00061
00067     texture::s_ptr get_texture(string_ref name);
00068
00074     bool add_texture(texture::s_ptr product);
00075
00082     c_data get_shader(string_ref name,
00083                       bool reload = false);
00084
00090     c_data reload_shader(string_ref name) {
00091         return get_shader(name, true);
00092     }
00093
00101     data compile_shader(c_data product,
00102                        string_ref name,
00103                        string_ref filename) const;
00104
00108     void destroy();
00109
00113     void clear();
00114
00116     id_registry<mesh, string> meshes;
00117
00119     id_registry<texture, string> textures;
00120
00124     enum shader_optimization : index {
00125         none = 0,
00126         size,
00127         performance
00128     };
00129
00131     shader_optimization shader_opt = shader_optimization::performance;
00132
00136     enum shader_language : index {
00137         glsl = 0,
00138         hlsl
00139     };
00140
00142     shader_language shader_lang = shader_language::glsl;
00143
00145     bool shader_debug = false;
00146
00147 private:
00153     void update_hash(string_ref name,
00154                     string_map_ref file_hash_map) const;
00155
00161     bool valid_shader(string_ref name) const;
00162
00164     using shader_map = std::map<string, data>;
00165
00167     shader_map m_shaders;
00168 };
00169
00170 } // namespace lava

```



## 5.87 liblava/engine/props.hpp File Reference

Props.

```
#include "liblava/file/file_utils.hpp"
#include "liblava/file/json.hpp"
#include "liblava/frame/argh.hpp"
#include "liblava/fwd.hpp"
```

### Classes

- struct [lava::props](#)  
*Props.*
- struct [lava::props::item](#)  
*Prop item.*

### 5.87.1 Detailed Description

Props.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.88 props.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/file/file_utils.hpp"
00011 #include "liblava/file/json.hpp"
00012 #include "liblava/frame/argh.hpp"
00013 #include "liblava/fwd.hpp"
00014
00015 namespace lava {
00016
00020 struct props : configurable {
00022     engine* app = nullptr;
00023
00027     struct item {
00029         using map = std::map<string, item>;
00030
00035         item(string_ref filename)
00036             : filename(filename) {}
00037
00039         string filename;
00040
00042         file_data data;
00043     };
00044
00050     void add(string_ref name,
00051             string_ref filename);
00052
00057     void remove(string_ref name);
```

```

00058
00065     bool install(string_ref name,
00066                 string_ref filename);
00067
00073     c_data operator()(string_ref name);
00074
00080     string_ref get_filename(string_ref name) const {
00081         return m_map.at(name).filename;
00082     }
00083
00089     void set_filename(string_ref name,
00090                     string_ref filename) {
00091         m_map.at(name).filename = filename;
00092     }
00093
00099     bool exists(string_ref name) const {
00100         return m_map.count(name);
00101     }
00102
00108     bool empty(string_ref name) const {
00109         return m_map.at(name).data.addr == nullptr;
00110     }
00111
00117     bool load(string_ref name);
00118
00123     void unload(string_ref name) {
00124         m_map.at(name).data = {};
00125     }
00126
00131     bool load_all();
00132
00136     void unload_all() {
00137         for (auto& [name, prop] : m_map)
00138             prop.data = {};
00139     }
00140
00145     bool check();
00146
00151     void parse(cmd_line cmd_line);
00152
00156     void clear() {
00157         m_map.clear();
00158     }
00159
00164     item::map const& get_all() const {
00165         return m_map;
00166     }
00167
00169     void set_json(json_ref j) override;
00170
00172     json get_json() const override;
00173
00174 private:
00176     item::map m_map;
00177 };
00178
00179 } // namespace lava

```

## 5.89 liblava/file.hpp File Reference

File module.

```

#include "liblava/file/file.hpp"
#include "liblava/file/file_system.hpp"
#include "liblava/file/file_utils.hpp"
#include "liblava/file/json.hpp"
#include "liblava/file/json_file.hpp"

```

### 5.89.1 Detailed Description

File module.

**Author**

Lava Block OÜ and contributors

**Copyright**

Copyright (c) 2018-present, MIT License

**5.90 file.hpp**

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/file/file.hpp"
00011 #include "liblava/file/file_system.hpp"
00012 #include "liblava/file/file_utils.hpp"
00013 #include "liblava/file/json.hpp"
00014 #include "liblava/file/json_file.hpp"
```

**5.91 liblava/file/file.hpp File Reference**

File access.

```
#include "liblava/core/data.hpp"
#include <fstream>
```

**Classes**

- struct [lava::file](#)

*File.*

**Enumerations**

- enum class [lava::file\\_type](#) : index { **none** = 0 , **fs** , **f\_stream** }

*File types.*

- enum class [lava::file\\_mode](#) : index { **read** = 0 , **write** }

*File modes.*

**Functions**

- bool [lava::file\\_error](#) (i64 result)

*Check file error result.*

**Variables**

- constexpr i64 const [lava::file\\_error\\_result](#) = undef

*File error result.*

### 5.91.1 Detailed Description

File access.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.91.2 Function Documentation

#### 5.91.2.1 file\_error()

```
bool lava::file_error (
    i64 result) [inline]
```

Check file error result.

#### Parameters

<i>result</i>	Result code to check
---------------	----------------------

#### Returns

Error result or okay

## 5.92 file.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/data.hpp"
00011 #include <fstream>
00012
00013 // fwd
00014 struct PHYSFS_File;
00015
00016 namespace lava {
00017
00021 enum class file_type : index {
00022     none = 0,
00023     fs,
00024     f_stream
00025 };
00026
00028 constexpr i64 const file_error_result = undef;
00029
00035 inline bool file_error(i64 result) {
00036     return result == file_error_result;
00037 }
00038
00042 enum class file_mode : index {
00043     read = 0,
00044     write
00045 };
00046
00050 struct file : no_copy_no_move {
```

```

00052     using ref = file const&;
00053
00059     explicit file(string_ref path = "",
00060                  file_mode mode = file_mode::read);
00061
00065     ~file();
00066
00073     bool open(string_ref path,
00074              file_mode mode = file_mode::read);
00075
00079     void close();
00080
00085     bool opened() const;
00086
00091     i64 get_size() const;
00092
00098     i64 read(data::ptr data) {
00099         return read(data,
00100                    to_ui64(get_size()));
00101     }
00102
00109     i64 read(data::ptr data, ui64 size);
00110
00117     i64 write(data::c_ptr data, ui64 size);
00118
00124     i64 seek(ui64 position);
00125
00130     i64 tell() const;
00131
00136     bool writable() const {
00137         return m_mode == file_mode::write;
00138     }
00139
00144     file_type get_type() const {
00145         return m_type;
00146     }
00147
00152     string_ref get_path() const {
00153         return m_path;
00154     }
00155
00156 private:
00158     file_type m_type = file_type::none;
00159
00161     file_mode m_mode = file_mode::read;
00162
00164     string m_path;
00165
00167     PHYSFS_File* m_file = nullptr;
00168
00170     mutable std::ifstream m_istream;
00171
00173     mutable std::ofstream m_ostream;
00174 };
00175
00176 } // namespace lava

```

## 5.93 liblava/file/file\_system.hpp File Reference

File system.

```

#include "liblava/core/version.hpp"
#include <filesystem>

```

### Classes

- struct [lava::file\\_system](#)

*File system.*

### 5.93.1 Detailed Description

File system.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.94 file\_system.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/version.hpp"
00011 #include <filesystem>
00012
00013 namespace lava {
00014
00018 struct file_system : no_copy_no_move {
00023     sem_version get_version();
00024
00029     string get_base_dir();
00030
00036     string get_full_base_dir(string_ref path);
00037
00042     string get_pref_dir();
00043
00048     string get_res_dir();
00049
00055     bool mount(string_ref path);
00056
00062     bool mount_base(string_ref base_dir_path);
00063
00069     bool exists(string_ref file);
00070
00076     string get_real_dir(string_ref file);
00077
00083     string_list enumerate_files(string_ref path);
00084
00093     bool initialize(string_ref argv_0,
00094                   string_ref org,
00095                   string_ref app,
00096                   string_ref ext);
00097
00101     void terminate();
00102
00107     string_list mount_res();
00108
00114     bool create_folder(string_ref name = "data");
00115
00119     void clean_pref_dir();
00120
00125     string_ref get_org() const {
00126         return m_org;
00127     }
00128
00133     string_ref get_app() const {
00134         return m_app;
00135     }
00136
00141     string_ref get_ext() const {
00142         return m_ext;
00143     }
00144
00149     bool ready() const {
00150         return m_initialized;
00151     }
00152
00153 private:

```

```

00155     bool m_initialized = false;
00156
00158     string m_org;
00159
00161     string m_app;
00162
00164     string m_ext;
00165
00167     string m_res_path;
00168 };
00169
00170 } // namespace lava

```

## 5.95 liblava/file/file\_utils.hpp File Reference

File utilities.

```
#include "liblava/core/data.hpp"
```

### Classes

- struct [lava::file\\_data](#)  
*File data.*
- struct [lava::file\\_delete](#)  
*File delete guard.*

### Functions

- bool [lava::read\\_file](#) (std::vector< char > &out, [string\\_ref](#) filename)  
*Read data from file.*
- bool [lava::write\\_file](#) ([string\\_ref](#) filename, char const \*data, [size\\_t](#) data\_size)  
*Write data to file.*
- bool [lava::extension](#) ([string\\_ref](#) filename, [string\\_ref](#) extension)  
*Check extension of file.*
- bool [lava::extension](#) ([string\\_ref](#) filename, [string\\_list\\_ref](#) extensions)  
*Check extensions of file.*
- [string](#) [lava::get\\_filename\\_from](#) ([string\\_ref](#) path, bool with\_extension=false)  
*Get the file name from path.*
- bool [lava::remove\\_existing\\_path](#) ([string](#) &target, [string\\_ref](#) path)  
*Remove existing path.*
- bool [lava::load\\_file\\_data](#) ([string\\_ref](#) filename, [data](#) &target)  
*Load file data.*

### 5.95.1 Detailed Description

File utilities.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.95.2 Function Documentation

### 5.95.2.1 `extension()` [1/2]

```
bool lava::extension (
    string_ref filename,
    string_list_ref extensions)
```

Check extensions of file.

#### Parameters

<i>filename</i>	Name of file
<i>extensions</i>	List of extensions to check

#### Returns

Extension found or not

### 5.95.2.2 `extension()` [2/2]

```
bool lava::extension (
    string_ref filename,
    string_ref extension)
```

Check extension of file.

#### Parameters

<i>filename</i>	Name of file
<i>extension</i>	Extension to check

#### Returns

Extension found or not

### 5.95.2.3 `get_filename_from()`

```
string lava::get_filename_from (
    string_ref path,
    bool with_extension = false)
```

Get the file name from path.

#### Parameters

<i>path</i>	Target path
<i>with_extension</i>	Include extension in file name

#### Returns

string File name



#### 5.95.2.4 load\_file\_data()

```
bool lava::load_file_data (
    string_ref filename,
    data & target)
```

Load file data.

Parameters

<i>filename</i>	Name of file
<i>target</i>	Target data

Returns

Load was successful or failed

#### 5.95.2.5 read\_file()

```
bool lava::read_file (
    std::vector< char > & out,
    string_ref filename)
```

Read data from file.

Parameters

<i>out</i>	File data
<i>filename</i>	Name of file

Returns

Read was successful or failed

#### 5.95.2.6 remove\_existing\_path()

```
bool lava::remove_existing_path (
    string & target,
    string_ref path)
```

Remove existing path.

Parameters

<i>target</i>	Target path
<i>path</i>	Path to remove

Returns

Remove was successful or failed

#### 5.95.2.7 write\_file()

```
bool lava::write_file (
    string_ref filename,
    char const * data,
    size_t data_size)
```

Write data to file.

**Parameters**

<i>filename</i>	Name of file
<i>data</i>	Data to write
<i>data_size</i>	Size of data

**Returns**

Write was successful or failed

**5.96 file\_utils.hpp**

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/data.hpp"
00011
00012 namespace lava {
00013
00020 bool read_file(std::vector<char>& out, string_ref filename);
00021
00029 bool write_file(string_ref filename,
00030                 char const* data,
00031                 size_t data_size);
00032
00039 bool extension(string_ref filename, string_ref extension);
00040
00047 bool extension(string_ref filename,
00048                 string_list_ref extensions);
00049
00056 string get_filename_from(string_ref path,
00057                           bool with_extension = false);
00058
00065 bool remove_existing_path(string& target,
00066                            string_ref path);
00067
00074 bool load_file_data(string_ref filename,
00075                      data& target);
00076
00080 struct file_data : u_data {
00082     using ref = file_data const&;
00083
00085     using u_data::u_data;
00086
00091     explicit file_data(string_ref filename)
00092     : filename(filename) {
00093         load_file_data(filename, *this);
00094     }
00095
00097     string filename;
00098 };
00099
00103 struct file_delete : no_copy_no_move {
00108     explicit file_delete(string filename = "")
00109     : filename(filename) {}
00110
00114     ~file_delete();
00115
00117     string filename;
00118
00120     bool active = true;
00121 };
00122
00123 } // namespace lava

```

**5.97 json.hpp**

```

00001
00008 #pragma once
00009

```

```

00010 #include "liblava/core/types.hpp"
00011 #include "nlohmann/json.hpp"
00012
00013 namespace lava {
00014
00016 using json = nlohmann::json;
00017
00019 using json_ref = json const&;
00020
00024 struct configurable : interface {
00029     virtual void set_json(json_ref j) = 0;
00030
00035     virtual json get_json() const = 0;
00036 };
00037
00038 } // namespace lava

```

## 5.98 liblava/file/json\_file.hpp File Reference

Json file.

```
#include "liblava/file/json.hpp"
```

### Classes

- struct [lava::json\\_file](#)  
*Json file.*
- struct [lava::json\\_file::callback](#)  
*Json file callback.*

### 5.98.1 Detailed Description

Json file.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.99 json\_file.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/file/json.hpp"
00011
00012 namespace lava {
00013
00017 struct json_file {
00022     explicit json_file(string_ref path = "config.json");
00023
00027     struct callback {
00029         using list = std::vector<callback*>;

```

```

00030
00032     using load_func = std::function<void(json_ref)>;
00033
00035     load_func on_load;
00036
00038     using save_func = std::function<json()>;
00039
00041     save_func on_save;
00042 };
00043
00048 void add(callback* callback);
00049
00054 void remove(callback* callback);
00055
00059 void clear() {
00060     m_callbacks.clear();
00061 }
00062
00067 void set(string_ref value) {
00068     m_path = value;
00069 }
00070
00075 string_ref get() const {
00076     return m_path;
00077 }
00078
00083 bool load();
00084
00089 bool save();
00090
00091 private:
00093     string m_path;
00094
00096     callback::list m_callbacks;
00097 };
00098
00099 } // namespace lava

```

## 5.100 liblava/frame.hpp File Reference

Frame module.

```

#include "liblava/frame/argh.hpp"
#include "liblava/frame/driver.hpp"
#include "liblava/frame/frame.hpp"
#include "liblava/frame/gamepad.hpp"
#include "liblava/frame/input.hpp"
#include "liblava/frame/render_target.hpp"
#include "liblava/frame/renderer.hpp"
#include "liblava/frame/swapchain.hpp"
#include "liblava/frame/window.hpp"

```

### 5.100.1 Detailed Description

Frame module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.101 frame.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/frame/argh.hpp"
00011 #include "liblava/frame/driver.hpp"
00012 #include "liblava/frame/frame.hpp"
00013 #include "liblava/frame/gamepad.hpp"
00014 #include "liblava/frame/input.hpp"
00015 #include "liblava/frame/render_target.hpp"
00016 #include "liblava/frame/renderer.hpp"
00017 #include "liblava/frame/swapchain.hpp"
00018 #include "liblava/frame/window.hpp"
```

## 5.102 liblava/frame/frame.hpp File Reference

Framework.

```
#include "liblava/base/device.hpp"
#include "liblava/base/instance.hpp"
#include "liblava/base/platform.hpp"
#include "liblava/core/time.hpp"
#include "liblava/frame/argh.hpp"
#include "liblava/util/log.hpp"
#include "liblava/util/telegram.hpp"
```

### Classes

- struct [lava::frame\\_env](#)  
*Framework environment.*
- struct [lava::frame](#)  
*Framework.*

### Enumerations

- enum [lava::error](#) {  
  **not\_ready** = -1 , **create\_failed** = -2 , **init\_failed** = -3 , **load\_failed** = -4 ,  
  **run\_aborted** = -5 , **still\_running** = -6 , **program\_failed** = -7 }  
*Error codes.*

### Functions

- [ms lava::now](#) ()  
*Get the current time.*
- void [lava::handle\\_events](#) (bool wait=false)  
*Handle events.*
- void [lava::handle\\_events\\_timeout](#) (ms timeout)  
*Handle events with timeout.*
- void [lava::handle\\_events\\_timeout](#) (seconds timeout)  
*Handle events with timeout.*
- void [lava::post\\_empty\\_event](#) ()  
*Post an empty event.*

## Variables

- constexpr bool const **lava::run\_abort** = false  
*Run abort result.*
- constexpr bool const **lava::run\_continue** = true  
*Run continue result.*

## 5.102.1 Detailed Description

Framework.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.102.2 Function Documentation

### 5.102.2.1 handle\_events()

```
void lava::handle_events (
    bool wait = false)
```

Handle events.

#### Parameters

<i>wait</i>	Wait for events
-------------	-----------------

### 5.102.2.2 handle\_events\_timeout() [1/2]

```
void lava::handle_events_timeout (
    ms timeout)
```

Handle events with timeout.

#### Parameters

<i>timeout</i>	Wait timeout in milliseconds
----------------	------------------------------

### 5.102.2.3 handle\_events\_timeout() [2/2]

```
void lava::handle_events_timeout (
    seconds timeout)
```

Handle events with timeout.

## Parameters

<i>timeout</i>	Wait timeout in seconds
----------------	-------------------------

## 5.102.2.4 now()

```
ms lava::now ()
```

Get the current time.

## Returns

ms Current milliseconds

## 5.103 frame.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011 #include "liblava/base/instance.hpp"
00012 #include "liblava/base/platform.hpp"
00013 #include "liblava/core/time.hpp"
00014 #include "liblava/frame/argh.hpp"
00015 #include "liblava/util/log.hpp"
00016 #include "liblava/util/telegram.hpp"
00017
00018 namespace lava {
00019
00023 struct frame_env {
00025     using ref = frame_env const&;
00026
00030     explicit frame_env() {
00031         set_default();
00032     }
00033
00039     explicit frame_env(name app_name,
00040                        argh::parser cmd_line)
00041     : cmd_line(cmd_line) {
00042         info.app_name = app_name;
00043         set_default();
00044     }
00045
00049     void set_default();
00050
00052     argh::parser cmd_line;
00053
00055     log::config log;
00056
00058     instance_info info;
00059
00061     instance::create_param param;
00062
00064     instance::debug_config debug;
00065
00067     ui32 telegraph_thread_count = 4;
00068 };
00069
00073 enum error {
00074     not_ready = -1,
00075     create_failed = -2,
00076     init_failed = -3,
00077     load_failed = -4,
00078     run_aborted = -5,
00079     still_running = -6,
00080     program_failed = -7,
00081 };
00082
00087 ms now();
00088
```

```

00090 constexpr bool const run_abort = false;
00091
00093 constexpr bool const run_continue = true;
00094
00098 struct frame : interface, no_copy_no_move {
00100     using s_ptr = std::shared_ptr<frame>;
00101
00106     explicit frame(argh::parser cmd_line);
00107
00112     explicit frame(frame_env env);
00113
00117     ~frame() override;
00118
00123     bool ready() const {
00124         return m_initialized;
00125     }
00126
00128     using result = i32; // error < 0
00129
00134     result run();
00135
00140     bool shut_down();
00141
00143     using run_func = std::function<bool(id::ref)>;
00144
00146     using run_func_ref = run_func const&;
00147
00153     id add_run(run_func_ref func);
00154
00156     using run_end_func = std::function<void()>;
00157
00159     using run_end_func_ref = run_end_func const&;
00160
00166     id add_run_end(run_end_func_ref func);
00167
00169     using run_once_func = std::function<bool()>;
00170
00172     using run_once_func_ref = run_once_func const&;
00173
00178     void add_run_once(run_once_func_ref func) {
00179         m_run_once_list.push_back(func);
00180     }
00181
00187     bool remove(id::ref func_id);
00188
00193     ms get_running_time() const {
00194         return now() - m_start_time;
00195     }
00196
00201     r64 get_running_time_sec() const {
00202         return to_sec(get_running_time());
00203     }
00204
00209     cmd_line get_cmd_line() const {
00210         return m_env.cmd_line;
00211     }
00212
00217     frame_env::ref get_env() const {
00218         return m_env;
00219     }
00220
00225     name get_name() const {
00226         return m_env.info.app_name;
00227     }
00228
00233     bool waiting_for_events() const {
00234         return m_wait_for_events;
00235     }
00236
00241     void set_wait_for_events(bool value = true) {
00242         m_wait_for_events = value;
00243     }
00244
00246     lava::run_time run_time;
00247
00249     lava::platform platform;
00250
00252     message_dispatcher telegraph;
00253
00254 private:
00259     bool setup();
00260
00264     void teardown();
00265
00270     bool run_step();
00271
00275     void trigger_run_remove();

```



```

00276
00280     void trigger_run_end();
00281
00283     bool m_initialized = false;
00284
00286     frame_env m_env;
00287
00289     bool m_running = false;
00290
00292     bool m_wait_for_events = false;
00293
00295     ms m_start_time;
00296
00298     using run_func_map = std::map<id, run_func>;
00299
00301     run_func_map m_run_map;
00302
00304     using run_end_func_map = std::map<id, run_end_func>;
00305
00307     run_end_func_map m_run_end_map;
00308
00310     using run_once_func_list = std::vector<run_once_func>;
00311
00313     run_once_func_list m_run_once_list;
00314
00316     id::list m_run_remove_list;
00317 };
00318
00323 void handle_events(bool wait = false);
00324
00329 void handle_events_timeout(ms timeout);
00330
00335 void handle_events_timeout(seconds timeout);
00336
00340 void post_empty_event();
00341
00342 } // namespace lava

```

## 5.104 liblava/frame/argh.hpp File Reference

Json.

```

#include "argh.h"
#include "liblava/core/types.hpp"

```

### Typedefs

- using **lava::cmd\_line** = argh::parser const&  
*Command line.*

### Functions

- void **lava::log\_command\_line** (cmd\_line cmd\_line)  
*Log command line.*
- **string** **lava::get\_cmd** (cmd\_line cmd\_line, std::initializer\_list< **name** const > **names**)  
*Get the value from command line arguments.*

### 5.104.1 Detailed Description

Json.

Command line arguments.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.104.2 Function Documentation

#### 5.104.2.1 `get_cmd()`

```
string lava::get_cmd (
    cmd_line cmd_line,
    std::initializer_list< name const > names)
```

Get the value from command line arguments.

#### Parameters

<i>cmd_line</i>	Command line arguments
<i>names</i>	Argument names

#### Returns

string Value of command line argument

#### 5.104.2.2 `log_command_line()`

```
void lava::log_command_line (
    cmd_line cmd_line)
```

Log command line.

#### Parameters

<i>cmd_line</i>	Command line arguments
-----------------	------------------------

## 5.105 argh.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "argh.h"
00011 #include "liblava/core/types.hpp"
00012
00013 namespace lava {
00014
00016 using cmd_line = argh::parser const&;
00017
00022 void log_command_line(cmd_line cmd_line);
00023
00030 string get_cmd(cmd_line cmd_line,
00031               std::initializer_list<name const> names);
00032
00033 } // namespace lava
```

## 5.106 liblava/frame/driver.hpp File Reference

Stage driver.

```
#include "liblava/frame/argh.hpp"
```

### Classes

- struct [lava::stage](#)  
*Stage.*
- struct [lava::driver](#)  
*Stage driver.*
- struct [lava::driver::result](#)  
*Driver result.*

### Macros

- #define **STAGE\_OBJ** stage\_  
*Stage object.*
- #define **STAGE\_FUNC** \_stage  
*Stage function.*
- #define **STR**\_(n, m)  
*String concatenation.*
- #define **STR**(n, m)  
*String concatenation.*
- #define **LAVA\_STAGE**(ID, NAME)  
*lava stage macro*

### 5.106.1 Detailed Description

Stage driver.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.106.2 Macro Definition Documentation

#### 5.106.2.1 LAVA\_STAGE

```
#define LAVA_STAGE(  
    ID,  
    NAME)
```

##### Value:

```
lava::i32 STR(STAGE_FUNC, ID)(argh::parser argh); \  
lava::stage STR(STAGE_OBJ, ID)(ID, NAME, ::STR(STAGE_FUNC, ID)); \  
lava::i32 STR(STAGE_FUNC, ID)(argh::parser argh)
```

lava stage macro

#### 5.106.2.2 STR

```
#define STR(  
    n,  
    m)
```

##### Value:

```
STR_(n, m)
```

String concatenation.

#### 5.106.2.3 STR\_

```
#define STR_(  
    n,  
    m)
```

##### Value:

```
n##m
```

String concatenation.

## 5.107 driver.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/frame/argh.hpp"
00011
00012 namespace lava {
00013
00017 struct stage {
00019     using map = std::map<index, stage*>;
00020
00022     using func = std::function<i32(argh::parser)>;
00023
00030     explicit stage(ui32 id,
00031                   string_ref name,
00032                   func func);
00033
00035     index id = 0;
00036
00038     string name;
00039
00041     func on_func;
00042 };
00043
00047 struct driver {
00051     enum error {
00052         stages_empty = -1,
00053         stage_not_found = -2,
00054         undef_run = -3,
00055     };
00056
00060     struct result {
00062         i32 driver = 0;
00063
00065         i32 selected = 0;
00066     };
00067
00072     static driver& instance() {
00073         static driver driver;
00074         return driver;
00075     }
00076
00081     void add_stage(stage* stage) {
00082         assert(!m_stages.count(stage->id) && "stage id already defined.");
00083         m_stages.emplace(stage->id, stage);
00084     }
00085
00090     stage::map const& get_stages() const {
00091         return m_stages;
00092     }
00093
00099     i32 run(argh::parser cmd_line = {});
00100
00102     using run_func = std::function<result(argh::parser)>;
00103
00105     run_func on_run;
00106
00107 private:
00111     driver() = default;
00112
00114     stage::map m_stages;
00115 };
00116
00117 } // namespace lava
00118
00120 #define STAGE_OBJ stage_
00121
00123 #define STAGE_FUNC _stage
00124
00126 #define STR_(n, m) n##m
00127
00129 #define STR(n, m) STR_(n, m)
00130
00132 #define LAVA_STAGE(ID, NAME) \
00133     lava::i32 STR(STAGE_FUNC, ID)(argh::parser argh); \
00134     lava::stage STR(STAGE_OBJ, ID)(ID, NAME, ::STR(STAGE_FUNC, ID)); \
00135     lava::i32 STR(STAGE_FUNC, ID)(argh::parser argh)

```

## 5.108 liblava/frame/gamepad.hpp File Reference

Gamepad manager.

```
#include "liblava/core/id.hpp"
```

### Classes

- struct [lava::gamepad](#)  
*Gamepad.*
- struct [lava::gamepad\\_manager](#)  
*Gamepad manager.*

### Typedefs

- using [lava::gamepad\\_id\\_ref](#) = [gamepad\\_id](#) const&  
*Reference to gamepad id.*
- using [lava::gamepad\\_button\\_ref](#) = [gamepad\\_button](#) const&  
*Reference to gamepad button.*
- using [lava::gamepad\\_axis\\_ref](#) = [gamepad\\_axis](#) const&  
*Reference to gamepad axis.*

### Enumerations

- enum class [lava::gamepad\\_id](#) : index {  
    [\\_1](#) = 0 , [\\_2](#) , [\\_3](#) , [\\_4](#) ,  
    [\\_5](#) , [\\_6](#) , [\\_7](#) , [\\_8](#) ,  
    [\\_9](#) , [\\_10](#) , [\\_11](#) , [\\_12](#) ,  
    [\\_13](#) , [\\_14](#) , [\\_15](#) , [\\_16](#) ,  
    [last](#) = [\\_16](#) }  
*Gamepad ids.*
- enum class [lava::gamepad\\_button](#) : index {  
    [a](#) = 0 , [b](#) , [x](#) , [y](#) ,  
    [left\\_bumper](#) , [right\\_bumper](#) , [back](#) , [start](#) ,  
    [guide](#) , [left\\_thumb](#) , [right\\_thumb](#) , [dpad\\_up](#) ,  
    [dpad\\_right](#) , [dpad\\_down](#) , [dpad\\_left](#) , [last](#) = [dpad\\_left](#) ,  
    [cross](#) = [a](#) , [circle](#) = [b](#) , [square](#) = [x](#) , [triangle](#) = [y](#) }  
*Gamepad buttons.*
- enum class [lava::gamepad\\_axis](#) : index {  
    [left\\_x](#) = 0 , [left\\_y](#) , [right\\_x](#) , [right\\_y](#) ,  
    [left\\_trigger](#) , [right\\_trigger](#) , [last](#) = [right\\_trigger](#) }  
*Gamepad axis.*

### Functions

- [gamepad::list](#) [lava::gamepads](#) ()  
*Get list of all gamepads.*

### 5.108.1 Detailed Description

Gamepad manager.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.108.2 Function Documentation

#### 5.108.2.1 gamepads()

```
gamepad::list lava::gamepads ()
```

Get list of all gamepads.

#### Returns

gamepad::list List of gamepads

## 5.109 gamepad.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/id.hpp"
00011
00012 namespace lava {
00013
00017 enum class gamepad_id : index {
00018     _1 = 0,
00019     _2,
00020     _3,
00021     _4,
00022     _5,
00023     _6,
00024     _7,
00025     _8,
00026     _9,
00027     _10,
00028     _11,
00029     _12,
00030     _13,
00031     _14,
00032     _15,
00033     _16,
00034
00035     last = _16,
00036 };
00037
00039 using gamepad_id_ref = gamepad_id const&;
00040
00044 enum class gamepad_button : index {
00045     a = 0,
00046     b,
00047     x,
00048     y,
00049
00050     left_bumper,
```

```

00051     right_bumper,
00052
00053     back,
00054     start,
00055     guide,
00056
00057     left_thumb,
00058     right_thumb,
00059
00060     dpad_up,
00061     dpad_right,
00062     dpad_down,
00063     dpad_left,
00064
00065     last = dpad_left,
00066
00067     cross = a,
00068     circle = b,
00069     square = x,
00070     triangle = y,
00071 };
00072
00073 using gamepad_button_ref = gamepad_button const&;
00074
00075 enum class gamepad_axis : index {
00076     left_x = 0,
00077     left_y,
00078
00079     right_x,
00080     right_y,
00081
00082     left_trigger,
00083     right_trigger,
00084
00085     last = right_trigger,
00086 };
00087
00088 using gamepad_axis_ref = gamepad_axis const&;
00089
00090 struct gamepad {
00091     using list = std::vector<gamepad>;
00092
00093     using ref = gamepad const&;
00094
00095     explicit gamepad(gamepad_id_ref pad_id = gamepad_id::_1);
00096
00097     bool ready() const;
00098
00099     bool update();
00100
00101     bool pressed(gamepad_button_ref button) const {
00102         return m_state.buttons[to_ui32(button)];
00103     }
00104
00105     r32 value(gamepad_axis_ref axis) const {
00106         return m_state.axes[to_ui32(axis)];
00107     }
00108
00109     gamepad_id_ref get_pad_id() const {
00110         return m_pad_id;
00111     }
00112
00113     ui32 get_id() const {
00114         return to_ui32(get_pad_id());
00115     }
00116
00117     name get_name() const;
00118
00119 private:
00120     gamepad_id m_pad_id;
00121
00122     struct state {
00123         uchar buttons[15];
00124
00125         r32 axes[6];
00126     };
00127
00128     gamepad::state m_state;
00129 };
00130
00131 gamepad::list gamepads();
00132
00133 struct gamepad_manager {
00134     using listener_func = std::function<bool(gamepad, bool)>;
00135
00136     static gamepad_manager& singleton() {
00137         static gamepad_manager manager;

```



```

00201         return manager;
00202     }
00203
00209     id add(listener_func listener);
00210
00215     void remove(id::ref func_id);
00216
00217 private:
00221     explicit gamepad_manager();
00222
00224     using listener_map = std::map<id, listener_func>;
00225
00227     listener_map m_map;
00228 };
00229
00230 } // namespace lava

```

## 5.110 liblava/frame/input.hpp File Reference

Input handling.

```

#include "liblava/core/id.hpp"
#include "liblava/core/misc.hpp"

```

### Classes

- struct [lava::key\\_event](#)  
*Key event.*
- struct [lava::scroll\\_offset](#)  
*Input scroll offset.*
- struct [lava::scroll\\_event](#)  
*Scroll event.*
- struct [lava::mouse\\_position](#)  
*Input mouse position.*
- struct [lava::mouse\\_move\\_event](#)  
*Mouse move event.*
- struct [lava::mouse\\_button\\_event](#)  
*Mouse button event.*
- struct [lava::path\\_drop\\_event](#)  
*Path drop event.*
- struct [lava::mouse\\_active\\_event](#)  
*Mouse active event.*
- struct [lava::input\\_callback](#)  
*Input callback.*
- struct [lava::input\\_events< T >](#)  
*List of input events.*
- struct [lava::input](#)  
*Input handling.*
- struct [lava::tooltip](#)  
*Tooltip.*
- struct [lava::tooltip\\_list](#)  
*Tooltip list.*

## Typedefs

- using **lava::key\_ref** = **key** const&  
*Reference to key.*
- using **lava::keys** = std::vector<**key**>  
*List of keys.*
- using **lava::keys\_ref** = **keys** const&  
*Reference to list of keys.*
- using **lava::action\_ref** = **action** const&  
*Referenece to action.*
- using **lava::mod\_ref** = **mod** const&  
*Reference to mod.*
- using **lava::mouse\_position\_ref** = **mouse\_position** const&  
*Reference to mouse position.*
- using **lava::mouse\_button\_ref** = **mouse\_button** const&  
*Reference to mouse button.*
- using **lava::input\_key\_events** = **input\_events**<**key\_event**>  
*List of key events.*
- using **lava::input\_scroll\_events** = **input\_events**<**scroll\_event**>  
*List of scroll events.*
- using **lava::input\_mouse\_move\_events** = **input\_events**<**mouse\_move\_event**>  
*List of mouse move events.*
- using **lava::input\_mouse\_button\_events** = **input\_events**<**mouse\_button\_event**>  
*List of mouse button events.*
- using **lava::input\_mouse\_active\_events** = **input\_events**<**mouse\_active\_event**>  
*List of mouse active events.*
- using **lava::input\_path\_drop\_events** = **input\_events**<**path\_drop\_event**>  
*List of path drop events.*

## Enumerations

- enum class **lava::key** : i32 {  
**unknown** = undef , **space** = 32 , **apostrophe** = 29 , **comma** = 44 ,  
**minus** = 45 , **period** = 46 , **slash** = 47 , **\_0** = 48 ,  
**\_1** , **\_2** , **\_3** , **\_4** ,  
**\_5** , **\_6** , **\_7** , **\_8** ,  
**\_9** , **semicolon** = 59 , **equal** = 61 , **a** = 65 ,  
**b** , **c** , **d** , **e** ,  
**f** , **g** , **h** , **i** ,  
**j** , **k** , **l** , **m** ,  
**n** , **o** , **p** , **q** ,  
**r** , **s** , **t** , **u** ,  
**v** , **w** , **x** , **y** ,  
**z** , **left\_bracket** = 91 , **backslash** = 92 , **right\_bracket** = 93 ,  
**grave\_accent** = 96 , **world\_1** = 161 , **world\_2** = 162 , **escape** = 256 ,  
**enter** , **tab** , **backspace** , **insert** ,  
**del** , **right** , **left** , **down** ,  
**up** , **page\_up** , **page\_down** , **home** ,  
**end** , **caps\_lock** = 280 , **scroll\_lock** , **num\_lock** ,  
**print\_screen** , **pause** , **f1** = 290 , **f2** ,  
**f3** , **f4** , **f5** , **f6** ,  
**f7** , **f8** , **f9** , **f10** ,  
**f11** , **f12** , **f13** , **f14** ,

```

f15 , f16 , f17 , f18 ,
f19 , f20 , f21 , f22 ,
f23 , f24 , f25 , kp_0 = 320 ,
kp_1 , kp_2 , kp_3 , kp_4 ,
kp_5 , kp_6 , kp_7 , kp_8 ,
kp_9 , kp_decimal , kp_divide , kp_multiply ,
kp_subtract , kp_add , kp_enter , kp_equal ,
left_shift = 340 , left_control , left_alt , left_super ,
right_shift , right_control , right_alt , right_super ,
menu , last = menu }

```

*Input keys.*

- enum class [lava::action](#) : index { **release** = 0 , **press** , **repeat** }

*Input actions.*

- enum class [lava::mod](#) : flag {  
**none** = 0 << 0 , **shift** = 1 << 0 , **control** = 1 << 1 , **alt** = 1 << 2 ,  
**super** = 1 << 3 , **caps\_lock** = 1 << 4 , **num\_lock** = 1 << 5 }

*Input mods.*

- enum class [lava::mouse\\_button](#) : index {  
**\_1** = 0 , **\_2** , **\_3** , **\_4** ,  
**\_5** , **\_6** , **\_7** , **\_8** ,  
**last** = **\_8** , **left** = **\_1** , **right** = **\_2** , **middle** = **\_3** }

*Input mouse buttons.*

## Functions

- [string lava::to\\_string](#) (key k)  
Convert input key to string.
- [i32 lava::get\\_scancode](#) (key key)  
Get scancode based on key.
- [bool lava::check\\_mod](#) (mod m, mod c)  
Check if mod is active.
- [string lava::to\\_string](#) (mod m)  
Convert input mod to string.

## Variables

- constexpr bool const **lava::input\_ignore** = false  
Input ignore result.
- constexpr bool const **lava::input\_done** = true  
Input done result.

### 5.110.1 Detailed Description

Input handling.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.110.2 Function Documentation

### 5.110.2.1 `check_mod()`

```
bool lava::check_mod (
    mod m,
    mod c) [inline]
```

Check if mod is active.

#### Parameters

<i>m</i>	Target mod
<i>c</i>	Mod to check

#### Returns

Mod is active or not

### 5.110.2.2 `get_scancode()`

```
i32 lava::get_scancode (
    key key)
```

Get scancode based on key.

#### Parameters

<i>key</i>	Input key
------------	-----------

#### Returns

i32 Input scan code

### 5.110.2.3 `to_string()` [1/2]

```
string lava::to_string (
    key k)
```

Convert input key to string.

#### Parameters

<i>k</i>	Input key
----------	-----------

#### Returns

string String representation

### 5.110.2.4 `to_string()` [2/2]

```
string lava::to_string (
    mod m)
```

Convert input mod to string.

## Parameters

<i>m</i>	Input mod
----------	-----------

## Returns

string String representation

## 5.111 input.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/id.hpp"
00011 #include "liblava/core/misc.hpp"
00012
00013 namespace lava {
00014
00018 enum class key : i32 {
00019     unknown = undef,
00020
00021     /* printable keys */
00022
00023     space = 32,
00024     apostrophe = 29, /* ' */
00025     comma = 44,      /* , */
00026     minus = 45,      /* - */
00027     period = 46,     /* . */
00028     slash = 47,      /* / */
00029
00030     _0 = 48,
00031     _1,
00032     _2,
00033     _3,
00034     _4,
00035     _5,
00036     _6,
00037     _7,
00038     _8,
00039     _9,
00040
00041     semicolon = 59, /* ; */
00042     equal = 61,     /* = */
00043
00044     a = 65,
00045     b,
00046     c,
00047     d,
00048     e,
00049     f,
00050     g,
00051     h,
00052     i,
00053     j,
00054     k,
00055     l,
00056     m,
00057     n,
00058     o,
00059     p,
00060     q,
00061     r,
00062     s,
00063     t,
00064     u,
00065     v,
00066     w,
00067     x,
00068     y,
00069     z,
00070
00071     left_bracket = 91, /* [ */
00072     backslash = 92,   /* \ */
00073     right_bracket = 93, /* ] */
00074     grave_accent = 96, /* ` */

```

```
00075
00076     world_1 = 161, /* non-US #1 */
00077     world_2 = 162, /* non-US #2 */
00078
00079     /* function keys */
00080
00081     escape = 256,
00082     enter,
00083     tab,
00084     backspace,
00085     insert,
00086     del, /* delete */
00087
00088     right,
00089     left,
00090     down,
00091     up,
00092
00093     page_up,
00094     page_down,
00095     home,
00096     end,
00097
00098     caps_lock = 280,
00099     scroll_lock,
00100     num_lock,
00101     print_screen,
00102     pause,
00103
00104     f1 = 290,
00105     f2,
00106     f3,
00107     f4,
00108     f5,
00109     f6,
00110     f7,
00111     f8,
00112     f9,
00113     f10,
00114     f11,
00115     f12,
00116     f13,
00117     f14,
00118     f15,
00119     f16,
00120     f17,
00121     f18,
00122     f19,
00123     f20,
00124     f21,
00125     f22,
00126     f23,
00127     f24,
00128     f25,
00129
00130     kp_0 = 320,
00131     kp_1,
00132     kp_2,
00133     kp_3,
00134     kp_4,
00135     kp_5,
00136     kp_6,
00137     kp_7,
00138     kp_8,
00139     kp_9,
00140
00141     kp_decimal,
00142     kp_divide,
00143     kp_multiply,
00144     kp_subtract,
00145     kp_add,
00146     kp_enter,
00147     kp_equal,
00148
00149     left_shift = 340,
00150     left_control,
00151     left_alt,
00152     left_super,
00153
00154     right_shift,
00155     right_control,
00156     right_alt,
00157     right_super,
00158
00159     menu,
00160
00161     last = menu,
```

```

00162 };
00163
00165 using key_ref = key const&;
00166
00168 using keys = std::vector<key>;
00169
00171 using keys_ref = keys const&;
00172
00178 string to_string(key k);
00179
00185 i32 get_scancode(key key);
00186
00190 enum class action : index {
00191     release = 0,
00192     press,
00193     repeat
00194 };
00195
00197 using action_ref = action const&;
00198
00202 enum class mod : flag {
00203     none = 0 << 0,
00204     shift = 1 << 0,
00205     control = 1 << 1,
00206     alt = 1 << 2,
00207     super = 1 << 3,
00208     caps_lock = 1 << 4,
00209     num_lock = 1 << 5,
00210 };
00211
00212 ENUM_FLAG_OPERATORS(mod)
00213
00214
00220 inline bool check_mod(mod m, mod c) {
00221     return (m & c) != mod::none;
00222 }
00223
00225 using mod_ref = mod const&;
00226
00232 string to_string(mod m);
00233
00237 struct key_event {
00239     using ref = key_event const&;
00240
00242     using func = std::function<bool(ref)>;
00243
00245     using listeners = std::map<id, func>;
00246
00248     using list = std::vector<key_event>;
00249
00251     id sender;
00252
00254     lava::key key;
00255
00257     lava::action action;
00258
00260     lava::mod mod;
00261
00263     i32 scancode = 0;
00264
00270     bool pressed(key_ref k) const {
00271         return (action == action::press) && (key == k);
00272     }
00273
00279     bool released(key_ref k) const {
00280         return (action == action::release) && (key == k);
00281     }
00282
00288     bool repeated(key_ref k) const {
00289         return (action == action::repeat) && (key == k);
00290     }
00291
00296     bool active() const {
00297         return (action == action::press) || (action == action::repeat);
00298     }
00299
00306     bool pressed(key_ref k, mod_ref m) const {
00307         return pressed(k) && (mod == m);
00308     }
00309 };
00310
00314 struct scroll_offset {
00316     r64 x = 0.0;
00317
00319     r64 y = 0.0;
00320 };
00321

```

```

00325 struct scroll_event {
00327     using ref = scroll_event const&;
00328
00330     using func = std::function<bool(ref)>;
00331
00333     using listeners = std::map<id, func>;
00334
00336     using list = std::vector<scroll_event>;
00337
00339     id sender;
00340
00342     scroll_offset offset;
00343 };
00344
00348 struct mouse_position {
00350     r64 x = 0.0;
00351
00353     r64 y = 0.0;
00354 };
00355
00357 using mouse_position_ref = mouse_position const&;
00358
00362 struct mouse_move_event {
00364     using ref = mouse_move_event const&;
00365
00367     using func = std::function<bool(ref)>;
00368
00370     using listeners = std::map<id, func>;
00371
00373     using list = std::vector<mouse_move_event>;
00374
00376     id sender;
00377
00379     mouse_position position;
00380 };
00381
00385 enum class mouse_button : index {
00386     _1 = 0,
00387     _2,
00388     _3,
00389     _4,
00390     _5,
00391     _6,
00392     _7,
00393     _8,
00394
00395     last = _8,
00396
00397     left = _1,
00398     right = _2,
00399     middle = _3,
00400 };
00401
00403 using mouse_button_ref = mouse_button const&;
00404
00408 struct mouse_button_event {
00410     using ref = mouse_button_event const&;
00411
00413     using func = std::function<bool(ref)>;
00414
00416     using listeners = std::map<id, func>;
00417
00419     using list = std::vector<mouse_button_event>;
00420
00422     id sender;
00423
00425     mouse_button button;
00426
00428     lava::action action;
00429
00431     lava::mod mod;
00432
00438     bool pressed(mouse_button_ref b) const {
00439         return action == action::press && button == b;
00440     }
00441
00447     bool released(mouse_button_ref b) const {
00448         return action == action::release && button == b;
00449     }
00450 };
00451
00455 struct path_drop_event {
00457     using ref = path_drop_event const&;
00458
00460     using func = std::function<bool(ref)>;
00461
00463     using listeners = std::map<id, func>;

```



```

00464
00466     using list = std::vector<path_drop_event>;
00467
00469     id sender;
00470
00472     string_list files;
00473 };
00474
00478 struct mouse_active_event {
00480     using ref = mouse_active_event const&;
00481
00483     using func = std::function<bool(ref)>;
00484
00486     using listeners = std::map<id, func>;
00487
00489     using list = std::vector<mouse_active_event>;
00490
00492     id sender;
00493
00495     bool active = false;
00496 };
00497
00501 struct input_callback {
00503     using cptr = input_callback const*;
00504
00506     using list = std::vector<input_callback*>;
00507
00509     using clist = std::vector<cptr>;
00510
00515     template <typename T>
00516     using func = std::function<bool(typename T::ref)>;
00517
00519     key_event::func on_key_event;
00520
00522     scroll_event::func on_scroll_event;
00523
00525     mouse_move_event::func on_mouse_move_event;
00526
00528     mouse_button_event::func on_mouse_button_event;
00529
00531     mouse_active_event::func on_mouse_active_event;
00532
00534     path_drop_event::func on_path_drop_event;
00535 };
00536
00541 template <typename T>
00542 struct input_events : T::list {
00547     void add(typename T::ref event) {
00548         this->push_back(event);
00549     }
00550
00552     id_listeners<T> listeners;
00553 };
00554
00556 using input_key_events = input_events<key_event>;
00557
00559 using input_scroll_events = input_events<scroll_event>;
00560
00562 using input_mouse_move_events = input_events<mouse_move_event>;
00563
00565 using input_mouse_button_events = input_events<mouse_button_event>;
00566
00568 using input_mouse_active_events = input_events<mouse_active_event>;
00569
00571 using input_path_drop_events = input_events<path_drop_event>;
00572
00574 constexpr bool const input_ignore = false;
00575
00577 constexpr bool const input_done = true;
00578
00582 struct input {
00584     using ptr = input*;
00585
00587     input_key_events key;
00588
00590     input_scroll_events scroll;
00591
00593     input_mouse_move_events mouse_move;
00594
00596     input_mouse_button_events mouse_button;
00597
00599     input_mouse_active_events mouse_active;
00600
00602     input_path_drop_events path_drop;
00603
00607     void handle_events();
00608

```

```

00613     void add(input_callback::cptr callback) {
00614         m_callbacks.push_back(callback);
00615     }
00616
00621     void remove(input_callback::cptr callback) {
00622         lava::remove(m_callbacks, callback);
00623     }
00624
00629     mouse_position_ref get_mouse_position() const {
00630         return m_current_position;
00631     }
00632
00637     void set_mouse_position(mouse_position_ref position) {
00638         m_current_position = position;
00639     }
00640
00641 private:
00645     void handle_mouse_events();
00646
00648     mouse_position m_current_position;
00649
00651     input_callback::clist m_callbacks;
00652 };
00653
00657 struct tooltip {
00664     tooltip(string_ref name,
00665             key key,
00666             mod mod)
00667     : name(name), key(key), mod(mod) {
00668     }
00669
00671     using list = std::vector<tooltip>;
00672
00674     string name;
00675
00677     lava::key key;
00678
00680     lava::mod mod;
00681 };
00682
00686 struct tooltip_list {
00693     void add(string_ref name,
00694             key key,
00695             mod mod = mod::none) {
00696         m_tooltips.emplace_back(name, key, mod);
00697     }
00698
00702     void clear() {
00703         m_tooltips.clear();
00704     }
00705
00710     tooltip::list const& get_list() const {
00711         return m_tooltips;
00712     }
00713
00718     void set(tooltip::list const& list) {
00719         m_tooltips = list;
00720     }
00721
00726     string format_string() const;
00727
00728 private:
00730     tooltip::list m_tooltips;
00731 };
00732
00733 } // namespace lava

```

## 5.112 liblava/frame/render\_target.hpp File Reference

Render target.

```

#include "liblava/core/misc.hpp"
#include "liblava/frame/swapchain.hpp"
#include "liblava/fwd.hpp"
#include "liblava/resource/format.hpp"

```

## Classes

- struct [lava::render\\_target](#)

*Render target.*

## Functions

- [render\\_target::s\\_ptr lava::create\\_target](#) ([window \\*window](#), [device::ptr device](#), [bool v\\_sync=false](#), [bool triple\\_buffer=true](#), [surface\\_format\\_request request={}](#))  
*Create a new render target.*
- [render\\_target::s\\_ptr lava::create\\_target\\_v\\_sync](#) ([window \\*window](#), [device::ptr device](#), [surface\\_format\\_request request={}](#))  
*Create a new render target with V-Sync enabled.*
- [render\\_target::s\\_ptr lava::create\\_target\\_no\\_triple\\_buffer](#) ([window \\*window](#), [device::ptr device](#), [surface\\_format\\_request request={}](#))  
*Create a new render target that prefers VK\_PRESENT\_MODE\_IMMEDIATE\_KHR over VK\_PRESENT\_MODE\_MAILBOX\_KHR.*

### 5.112.1 Detailed Description

Render target.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.112.2 Function Documentation

#### 5.112.2.1 create\_target()

```
render_target::s_ptr lava::create_target (
    window * window,
    device::ptr device,
    bool v_sync = false,
    bool triple_buffer = true,
    surface_format_request request = {})
```

Create a new render target.

#### Parameters

<i>window</i>	Target window
<i>device</i>	Vulkan device
<i>v_sync</i>	V-Sync enabled
<i>triple_buffer</i>	VK_PRESENT_MODE_MAILBOX_KHR preferred over VK_PRESENT_MODE_IMMEDIATE_KHR
<i>request</i>	Surface format request

#### Returns

[render\\_target::s\\_ptr](#) Shared pointer to render target

### 5.112.2.2 create\_target\_no\_triple\_buffer()

```
render_target::s_ptr lava::create_target_no_triple_buffer (
    window * window,
    device::ptr device,
    surface_format_request request = {}) [inline]
```

Create a new render target that prefers VK\_PRESENT\_MODE\_IMMEDIATE\_KHR over VK\_PRESENT\_MODE\_MAILBOX\_KHR.

#### Parameters

<i>window</i>	Target window
<i>device</i>	Vulkan device
<i>request</i>	Surface format request

#### Returns

render\_target::s\_ptr Shared pointer to render target

### 5.112.2.3 create\_target\_v\_sync()

```
render_target::s_ptr lava::create_target_v_sync (
    window * window,
    device::ptr device,
    surface_format_request request = {}) [inline]
```

Create a new render target with V-Sync enabled.

#### Parameters

<i>window</i>	Target window
<i>device</i>	Vulkan device
<i>request</i>	Surface format request

#### Returns

render\_target::s\_ptr Shared pointer to render target

## 5.113 render\_target.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/misc.hpp"
00011 #include "liblava/frame/swapchain.hpp"
00012 #include "liblava/fwd.hpp"
00013 #include "liblava/resource/format.hpp"
00014
00015 namespace lava {
00016
00020 struct render_target : entity {
00022     using s_ptr = std::shared_ptr<render_target>;
```

```

00023
00028     static s_ptr make() {
00029         return std::make_shared<render_target>();
00030     }
00031
00042     bool create(device::ptr device,
00043                VkSurfaceKHR surface,
00044                VkSurfaceFormatKHR format,
00045                uv2 size,
00046                bool v_sync = false,
00047                bool triple_buffer = true);
00048
00052     void destroy();
00053
00058     uv2 get_size() const {
00059         return m_target.get_size();
00060     }
00061
00067     bool resize(uv2 new_size) {
00068         return m_target.resize(new_size);
00069     }
00070
00075     ui32 get_frame_count() const {
00076         return m_target.get_backbuffer_count();
00077     }
00078
00083     bool reload_request() const {
00084         return m_target.reload_request();
00085     }
00086
00090     void reload() {
00091         m_target.resize(m_target.get_size());
00092     }
00093
00098     device::ptr get_device() {
00099         return m_target.get_device();
00100     }
00101
00106     swapchain* get_swapchain() {
00107         return &m_target;
00108     }
00109
00114     VkFormat get_format() const {
00115         return m_target.get_format();
00116     }
00117
00122     image::s_list const& get_backbuffers() const {
00123         return m_target.get_backbuffers();
00124     }
00125
00131     inline image::s_ptr get_backbuffer(index index) {
00132         auto& backbuffers = get_backbuffers();
00133         if (index >= backbuffers.size())
00134             return nullptr;
00135
00136         return backbuffers.at(index);
00137     }
00138
00144     inline VkImage get_backbuffer_image(index index) {
00145         auto result = get_backbuffer(index);
00146         return result ? result->get() : 0;
00147     }
00148
00152     inline VkImage get_image(index index) {
00153         return get_backbuffer_image(index);
00154     }
00155
00160     void add_callback(target_callback::c_ptr callback) {
00161         m_target_callbacks.push_back(callback);
00162     }
00163
00168     void remove_callback(target_callback::c_ptr callback) {
00169         remove(m_target_callbacks, callback);
00170     }
00171
00173     using swapchain_start_func = std::function<bool()>;
00174
00176     swapchain_start_func on_swapchain_start;
00177
00179     using swapchain_stop_func = std::function<void()>;
00180
00182     swapchain_stop_func on_swapchain_stop;
00183
00185     using create_attachments_func = std::function<VkAttachments()>;
00186
00188     create_attachments_func on_create_attachments;
00189

```

```

00191     using destroy_attachments_func = std::function<void()>;
00192
00194     destroy_attachments_func on_destroy_attachments;
00195
00196 private:
00198     swapchain m_target;
00199
00201     swapchain::callback m_swapchain_callback;
00202
00204     target_callback::c_list m_target_callbacks;
00205 };
00206
00216 render_target::s_ptr create_target(window* window,
00217                                     device::ptr device,
00218                                     bool v_sync = false,
00219                                     bool triple_buffer = true,
00220                                     surface_format_request request = {});
00221
00229 inline render_target::s_ptr create_target_v_sync(window* window,
00230                                                    device::ptr device,
00231                                                    surface_format_request request = {}) {
00232     return create_target(window,
00233                          device,
00234                          true,
00235                          true,
00236                          request);
00237 }
00238
00246 inline render_target::s_ptr create_target_no_triple_buffer(window* window,
00247                                                            device::ptr device,
00248                                                            surface_format_request request = {}) {
00249     return create_target(window,
00250                          device,
00251                          false,
00252                          false,
00253                          request);
00254 }
00255
00256 } // namespace lava

```

## 5.114 liblava/frame/renderer.hpp File Reference

Plain renderer.

```

#include "liblava/frame/swapchain.hpp"
#include <optional>

```

### Classes

- struct [lava::renderer](#)  
*Plain renderer.*

### Typedefs

- using [lava::optional\\_index](#) = std::optional<[index](#)>  
*Optional frame index.*

### 5.114.1 Detailed Description

Plain renderer.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.115 renderer.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/frame/swapchain.hpp"
00011 #include <optional>
00012
00013 namespace lava {
00014
00016 using optional_index = std::optional<index>;
00017
00021 struct renderer : entity {
00023     using ptr = renderer*;
00024
00030     bool create(swapchain* target);
00031
00035     void destroy();
00036
00041     optional_index begin_frame();
00042
00048     bool end_frame(VkCommandBuffers const& cmd_buffers);
00049
00055     bool frame(VkCommandBuffers const& cmd_buffers) {
00056         if (!begin_frame())
00057             return false;
00058
00059         return end_frame(cmd_buffers);
00060     }
00061
00066     index get_frame() const {
00067         return m_current_frame;
00068     }
00069
00074     device::ptr get_device() {
00075         return m_device;
00076     }
00077
00079     VkSemaphores user_frame_wait_semaphores;
00080
00082     VkPipelineStageFlagsList user_frame_wait_stages;
00083
00085     VkSemaphores user_frame_signal_semaphores;
00086
00088     using destroy_func = std::function<void()>;
00089
00091     destroy_func on_destroy;
00092
00094     bool active = true;
00095
00096 private:
00098     device::ptr m_device = nullptr;
00099
00101     queue m_graphics_queue;
00102
00104     swapchain* m_target = nullptr;
00105
00107     index m_current_frame = 0;
00108
00110     ui32 m_queued_frames = 2;
00111
00113     ui32 m_current_sync = 0;
00114
00116     VkFences m_fences = {};
00117
00119     VkFences m_fences_in_use = {};
00120
00122     VkSemaphores m_image_acquired_semaphores = {};
00123
00125     VkSemaphores m_render_complete_semaphores = {};
00126 };
00127
00128 } // namespace lava

```

## 5.116 liblava/frame/swapchain.hpp File Reference

Swapchain.

```
#include "liblava/resource/image.hpp"
```

## Classes

- struct [lava::swapchain](#)  
*Swaphchain.*
- struct [lava::swapchain::callback](#)  
*Swapchain callback.*

### 5.116.1 Detailed Description

Swapchain.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.117 swapchain.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/resource/image.hpp"
00011
00012 namespace lava {
00013
00017 struct swapchain : entity {
00028     bool create(device::ptr device,
00029                 VkSurfaceKHR surface,
00030                 VkSurfaceFormatKHR format,
00031                 uv2 size,
00032                 bool v_sync = false,
00033                 bool triple_buffer = true);
00034
00038     void destroy();
00039
00045     bool resize(uv2 new_size);
00046
00050     void request_reload() {
00051         m_reload_request_active = true;
00052     }
00053
00058     bool reload_request() const {
00059         return m_reload_request_active;
00060     }
00061
00066     device::ptr get_device() {
00067         return m_device;
00068     }
00069
00074     uv2 get_size() const {
00075         return m_size;
00076     }
00077
00082     VkFormat get_format() const {
00083         return m_format.format;
00084     }
00085
00090     VkColorSpaceKHR get_color_space() const {
00091         return m_format.colorSpace;
00092     }
00093
00098     VkSwapchainKHR get() const {
00099         return m_vk_swapchain;
00100     }
}
```



```

00101
00106     ui32 get_backbuffer_count() const {
00107         return to_ui32(m_backbuffers.size());
00108     }
00109
00114     image::s_list const& get_backbuffers() const {
00115         return m_backbuffers;
00116     }
00117
00121     struct callback {
00123         using list = std::vector<callback*>;
00124
00126         using created_func = std::function<bool()>;
00127
00129         created_func on_created;
00130
00132         using destroyed_func = std::function<void()>;
00133
00135         destroyed_func on_destroyed;
00136     };
00137
00142     void add_callback(callback* cb);
00143
00148     void remove_callback(callback* cb);
00149
00154     bool v_sync() const {
00155         return m_v_sync_active;
00156     }
00157
00162     bool triple_buffer() const {
00163         return m_triple_buffer_active;
00164     }
00165
00171     bool surface_supported(index queue_family) const;
00172
00173 private:
00179     VkPresentModeKHR choose_present_mode(VkPresentModeKHRs const& present_modes) const;
00180
00186     VkSwapchainCreateInfoKHR create_info(VkPresentModeKHRs present_modes);
00187
00192     bool setup();
00193
00197     void teardown();
00198
00202     void destroy_backbuffer_views();
00203
00205     device::ptr m_device = nullptr;
00206
00208     VkSurfaceKHR m_surface = VK_NULL_HANDLE;
00209
00211     VkSurfaceFormatKHR m_format = {};
00212
00214     VkSwapchainKHR m_vk_swapchain = VK_NULL_HANDLE;
00215
00217     image::s_list m_backbuffers;
00218
00220     uv2 m_size;
00221
00223     bool m_reload_request_active = false;
00224
00226     bool m_v_sync_active = false;
00227
00229     bool m_triple_buffer_active = true;
00230
00232     callback::list m_callbacks;
00233 };
00234
00235 } // namespace lava

```

## 5.118 liblava/frame/window.hpp File Reference

Window.

```

#include "liblava/core/data.hpp"
#include "liblava/frame/input.hpp"
#include "liblava/util/math.hpp"
#include "vulkan/vulkan.h"
#include <optional>

```

## Classes

- struct [lava::window](#)  
*Window.*
- struct [lava::window::state](#)  
*Window state.*

## Functions

- VkSurfaceKHR [lava::create\\_surface](#) (GLFWwindow \*[window](#))  
*Create a new surface.*
- [window](#) \* [lava::get\\_window](#) (GLFWwindow \*[handle](#))  
*Get the window by GLFW handle.*

### 5.118.1 Detailed Description

Window.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.118.2 Function Documentation

#### 5.118.2.1 create\_surface()

```
VkSurfaceKHR lava::create_surface (
    GLFWwindow * window)
```

Create a new surface.

#### Parameters

<i>window</i>	GLFW window handle
---------------	--------------------

#### Returns

VkSurfaceKHR Vulkan surface

#### 5.118.2.2 get\_window()

```
window * lava::get_window (
    GLFWwindow * handle)
```

Get the window by GLFW handle.

## Parameters

<i>handle</i>	GLFW window handle
---------------	--------------------

## Returns

window\* Assigned Window

## 5.119 window.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/data.hpp"
00011 #include "liblava/frame/input.hpp"
00012 #include "liblava/util/math.hpp"
00013
00014 #define VK_NO_PROTOTYPES
00015 #include "vulkan/vulkan.h"
00016 #include <optional>
00017
00019 struct GLFWwindow;
00020
00021 namespace lava {
00022
00026 struct window : entity {
00030     struct state {
00032         using ref = state const&;
00033
00035         using optional = std::optional<window::state>;
00036
00040         explicit state() {}
00041
00043         i32 x = 0;
00044
00046         i32 y = 0;
00047
00049         ui32 width = 0;
00050
00052         ui32 height = 0;
00053
00055         bool fullscreen = false;
00056
00058         bool floating = false;
00059
00061         bool resizable = true;
00062
00064         bool decorated = true;
00065
00067         bool maximized = false;
00068
00070         index monitor = 0;
00071     };
00072
00074     using ptr = window*;
00075
00077     using s_ptr = std::shared_ptr<window>;
00078
00080     using event = std::function<void(s_ptr)>;
00081
00083     using s_map = std::map<id, s_ptr>;
00084
00086     using ref = window const&;
00087
00091     window() = default;
00092
00097     explicit window(name title)
00098     : m_title(title) {}
00099
00105     bool create(state::optional state = {});
00106
00110     void destroy();
00111
00116     state get_state() const;
00117
00122     void set_state(state& s);

```

```
00123
00128 void set_title(string_ref text);
00129
00134 string_ref get_title() const {
00135     return m_title;
00136 }
00137
00142 void set_save_name(string_ref save) {
00143     m_save_name = save;
00144 }
00145
00150 string_ref get_save_name() const {
00151     return m_save_name;
00152 }
00153
00159 void set_position(i32 x, i32 y);
00160
00166 void get_position(i32& x, i32& y) const;
00167
00173 void set_size(ui32 width, ui32 height);
00174
00180 void get_size(ui32& width, ui32& height) const;
00181
00187 void get_framebuffer_size(ui32& width, ui32& height) const;
00188
00193 uv2 get_size() const;
00194
00199 uv2 get_framebuffer_size() const;
00200
00206 void set_mouse_position(r64 x, r64 y);
00207
00213 void get_mouse_position(r64& x, r64& y) const;
00214
00219 v2 get_content_scale() const;
00220
00225 mouse_position get_mouse_position() const;
00226
00230 void hide_mouse_cursor();
00231
00235 void show_mouse_cursor();
00236
00241 r32 get_aspect_ratio() const;
00242
00246 void show();
00247
00251 void hide();
00252
00257 bool visible() const;
00258
00262 void iconify();
00263
00268 bool iconified() const;
00269
00273 void restore();
00274
00278 void maximize();
00279
00284 bool maximized() const;
00285
00289 void focus();
00290
00295 bool focused() const;
00296
00301 void set_fullscreen(bool active) {
00302     if (m_fullscreen_active != active)
00303         m_switch_mode_request_active = true;
00304 }
00305
00310 bool fullscreen() const {
00311     return m_fullscreen_active;
00312 }
00313
00318 bool hovered() const;
00319
00324 bool resizable() const;
00325
00330 void set_resizable(bool value);
00331
00336 bool decorated() const;
00337
00342 void set_decorated(bool value);
00343
00348 bool floating() const;
00349
00354 void set_floating(bool value);
00355
00360 bool close_request() const;
```

```

00361
00366     bool switch_mode_request() const {
00367         return m_switch_mode_request_active;
00368     }
00369
00375     bool switch_mode(state::optional state = {});
00376
00381     GLFWwindow* get() const {
00382         return m_handle;
00383     }
00384
00389     bool resize_request() const {
00390         return m_resize_request_active;
00391     }
00392
00397     bool handle_resize() {
00398         if (on_resize)
00399             if (!on_resize(m_framebuffer_width,
00400                           m_framebuffer_height))
00401                 return false;
00402
00403         m_resize_request_active = false;
00404         return true;
00405     }
00406
00410     void update_state() {
00411         get_position(m_pos_x, m_pos_y);
00412         get_size(m_width, m_height);
00413     }
00414
00416     using resize_func = std::function<bool(ui32, ui32)>;
00417
00419     resize_func on_resize;
00420
00425     void assign(input::ptr callback) {
00426         m_input = callback;
00427     }
00428
00433     void show_save_title(bool value = true) {
00434         m_save_title_active = value;
00435     }
00436
00441     bool save_title() const {
00442         return m_save_title_active;
00443     }
00444
00448     void update_title() {
00449         set_title(m_title);
00450     }
00451
00456     VkSurfaceKHR create_surface();
00457
00463     void set_icon(data::c_ptr data, uv2 size);
00464
00469     index detect_monitor() const;
00470
00474     void center();
00475
00476 private:
00480     void handle_message();
00481
00485     void handle_mouse_message();
00486
00488     GLFWwindow* m_handle = nullptr;
00489
00491     input::ptr m_input = nullptr;
00492
00494     string m_title = _lava_;
00495
00497     string m_save_name = _default_;
00498
00500     bool m_fullscreen_active = false;
00501
00503     bool m_save_title_active = false;
00504
00506     bool m_switch_mode_request_active = false;
00507
00509     bool m_resize_request_active = false;
00510
00512     ui32 m_framebuffer_width = 0;
00513
00515     ui32 m_framebuffer_height = 0;
00516
00518     i32 m_pos_x = 0;
00519
00521     i32 m_pos_y = 0;
00522

```

```

00524     ui32 m_width = 0;
00525
00527     ui32 m_height = 0;
00528 };
00529
00535 VkSurfaceKHR create_surface(GLFWwindow* window);
00536
00542 window* get_window(GLFWwindow* handle);
00543
00544 } // namespace lava

```

## 5.120 liblava/lava.hpp File Reference

All lava modules.

```

#include "liblava/app.hpp"
#include "liblava/asset.hpp"
#include "liblava/base.hpp"
#include "liblava/block.hpp"
#include "liblava/core.hpp"
#include "liblava/engine.hpp"
#include "liblava/file.hpp"
#include "liblava/frame.hpp"
#include "liblava/resource.hpp"
#include "liblava/util.hpp"

```

### 5.120.1 Detailed Description

All lava modules.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.121 lava.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/app.hpp"
00011 #include "liblava/asset.hpp"
00012 #include "liblava/base.hpp"
00013 #include "liblava/block.hpp"
00014 #include "liblava/core.hpp"
00015 #include "liblava/engine.hpp"
00016 #include "liblava/file.hpp"
00017 #include "liblava/frame.hpp"
00018 #include "liblava/resource.hpp"
00019 #include "liblava/util.hpp"

```

## 5.122 liblava/resource.hpp File Reference

Resource module.

```
#include "liblava/resource/buffer.hpp"
#include "liblava/resource/format.hpp"
#include "liblava/resource/image.hpp"
#include "liblava/resource/mesh.hpp"
#include "liblava/resource/texture.hpp"
```

### 5.122.1 Detailed Description

Resource module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.123 resource.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/resource/buffer.hpp"
00011 #include "liblava/resource/format.hpp"
00012 #include "liblava/resource/image.hpp"
00013 #include "liblava/resource/mesh.hpp"
00014 #include "liblava/resource/texture.hpp"
```

## 5.124 liblava/resource/buffer.hpp File Reference

Vulkan buffer.

```
#include "liblava/base/device.hpp"
```

#### Classes

- struct [lava::buffer](#)  
*Buffer.*

## Functions

- VkPipelineStageFlags [lava::buffer\\_usage\\_to\\_possible\\_stages](#) (VkBufferUsageFlags usage)  
*Get possible stages by bufferusage flags.*
- VkAccessFlags [lava::buffer\\_usage\\_to\\_possible\\_access](#) (VkBufferUsageFlags usage)  
*Get possible access by buffer usage flags.*

### 5.124.1 Detailed Description

Vulkan buffer.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.124.2 Function Documentation

#### 5.124.2.1 `buffer_usage_to_possible_access()`

```
VkAccessFlags lava::buffer_usage_to_possible_access (
    VkBufferUsageFlags usage)
```

Get possible access by buffer usage flags.

#### Parameters

<i>usage</i>	Buffer usage flags
--------------	--------------------

#### Returns

VkAccessFlags Access flags

#### 5.124.2.2 `buffer_usage_to_possible_stages()`

```
VkPipelineStageFlags lava::buffer_usage_to_possible_stages (
    VkBufferUsageFlags usage)
```

Get possible stages by bufferusage flags.

#### Parameters

<i>usage</i>	Buffer usage flags
--------------	--------------------

#### Returns

VkPipelineStageFlags Pipeline stage flags



## 5.125 buffer.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct buffer : entity {
00019     using s_ptr = std::shared_ptr<buffer>;
00020
00022     using s_list = std::vector<s_ptr>;
00023
00028     static s_ptr make() {
00029         return std::make_shared<buffer>();
00030     }
00031
00035     ~buffer() {
00036         destroy();
00037     }
00038
00052     bool create(device::ptr device,
00053                void const* data,
00054                size_t size,
00055                VkBufferUsageFlags usage,
00056                bool mapped = false,
00057                VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_GPU_ONLY,
00058                VkSharingMode sharing_mode = VK_SHARING_MODE_EXCLUSIVE,
00059                std::vector<ui32> const& shared_queue_family_indices = {},
00060                i32 alignment = undef);
00061
00074     bool create_mapped(device::ptr device,
00075                        void const* data,
00076                        size_t size,
00077                        VkBufferUsageFlags usage,
00078                        VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_CPU_TO_GPU,
00079                        VkSharingMode sharing_mode = VK_SHARING_MODE_EXCLUSIVE,
00080                        std::vector<ui32> const& shared_queue_family_indices = {},
00081                        i32 alignment = undef);
00082
00086     void destroy();
00087
00092     device::ptr get_device() {
00093         return m_device;
00094     }
00095
00100     bool valid() const {
00101         return m_vk_buffer != VK_NULL_HANDLE;
00102     }
00103
00108     VkBuffer get() const {
00109         return m_vk_buffer;
00110     }
00111
00116     VkDescriptorBufferInfo const& get_descriptor_info() const {
00117         return &m_descriptor;
00118     }
00119
00124     VkDeviceAddress get_address() const;
00125
00130     VkDeviceSize get_size() const {
00131         return m_allocation_info.size;
00132     }
00133
00138     void* get_mapped_data() const {
00139         return m_allocation_info.pMappedData;
00140     }
00141
00146     VkDeviceMemory get_device_memory() const {
00147         return m_allocation_info.deviceMemory;
00148     }
00149
00155     void flush(VkDeviceSize offset = 0,
00156                VkDeviceSize size = VK_WHOLE_SIZE);
00157
00162     VmaAllocation const& get_allocation() const {
00163         return m_allocation;
00164     }
00165
00170     VmaAllocationInfo const& get_allocation_info() const {
00171         return m_allocation_info;
00172     }
00173

```

```

00174 private:
00175     device::ptr m_device = nullptr;
00176
00177     VkBuffer m_vk_buffer = VK_NULL_HANDLE;
00178
00179     VmaAllocation m_allocation = nullptr;
00180
00181     VmaAllocationInfo m_allocation_info = {};
00182
00183     VkDescriptorBufferInfo m_descriptor = {};
00184 };
00185
00186 VkPipelineStageFlags buffer_usage_to_possible_stages(VkBufferUsageFlags usage);
00187
00188 VkAccessFlags buffer_usage_to_possible_access(VkBufferUsageFlags usage);
00189
00190 } // namespace lava

```

## 5.126 liblava/resource/format.hpp File Reference

Vulkan format.

```

#include "liblava/base/device.hpp"
#include <optional>

```

### Classes

- struct [lava::surface\\_format\\_request](#)  
*Surface format request.*

### Typedefs

- using [lava::VkFormat\\_optional](#) = std::optional<VkFormat>  
*Optional format.*

### Functions

- bool [lava::format\\_depth](#) (VkFormat format)  
*Check if format is depth compatible.*
- bool [lava::format\\_stencil](#) (VkFormat format)  
*Check if format is stencil compatible.*
- bool [lava::format\\_depth\\_stencil](#) (VkFormat format)  
*Check if format is depth or stencil compatible.*
- bool [lava::format\\_srgb](#) (VkFormat format)  
*Check if format is sRGB compatible.*
- bool [lava::format\\_bgr](#) (VkFormat format)  
*Check if format has BGR order.*
- VkImageAspectFlags [lava::format\\_aspect\\_mask](#) (VkFormat format)  
*Get image aspect mask of format.*
- void [lava::format\\_block\\_dim](#) (VkFormat format, [ui32](#) &width, [ui32](#) &height)  
*Get block dimension of format.*
- void [lava::format\\_align\\_dim](#) (VkFormat format, [ui32](#) &width, [ui32](#) &height)  
*Get align dimension of format.*
- void [lava::format\\_num\\_blocks](#) (VkFormat format, [ui32](#) &width, [ui32](#) &height)

- Get format number of blocks.*
- [ui32 lava::format\\_block\\_size](#) (VkFormat format, VkImageAspectFlags aspect)  
*Get format block size.*
- [ui32 lava::format\\_block\\_size](#) (VkFormat format)  
*Get format block size (with respective aspect mask)*
- [VkFormat\\_optional lava::find\\_supported\\_depth\\_format](#) (VkPhysicalDevice [physical\\_device](#))  
*Find the supported depth format.*
- [VkFormat\\_optional lava::find\\_supported\\_format](#) (VkPhysicalDevice [physical\\_device](#), [VkFormats](#) const &possible\_formats, VkImageUsageFlags usage)  
*Find the supported format.*
- [VkImageMemoryBarrier lava::image\\_memory\\_barrier](#) (VkImage [image](#), VkImageLayout old\_layout, VkImageLayout new\_layout)  
*Get image memory barrier.*
- void [lava::set\\_image\\_layout](#) (device::ptr device, VkCommandBuffer cmd\_buffer, VkImage [image](#), VkImageLayout old\_image\_layout, VkImageLayout new\_image\_layout, VkImageSubresourceRange subresource\_range, VkPipelineStageFlags src\_stage\_mask=VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT, VkPipelineStageFlags dst\_stage\_mask=VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT)  
*Set the image layout.*
- void [lava::set\\_image\\_layout](#) (device::ptr device, VkCommandBuffer cmd\_buffer, VkImage [image](#), VkImageAspectFlags aspect\_mask, VkImageLayout old\_image\_layout, VkImageLayout new\_image\_layout, VkPipelineStageFlags src\_stage\_mask=VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT, VkPipelineStageFlags dst\_stage\_mask=VK\_PIPELINE\_STAGE\_ALL\_COMMANDS\_BIT)  
*Set the image layout.*
- void [lava::insert\\_image\\_memory\\_barrier](#) (device::ptr device, VkCommandBuffer cmd\_buffer, VkImage [image](#), VkAccessFlags src\_access\_mask, VkAccessFlags dst\_access\_mask, VkImageLayout old\_image\_layout, VkImageLayout new\_image\_layout, VkPipelineStageFlags src\_stage\_mask, VkPipelineStageFlags dst\_stage\_mask, VkImageSubresourceRange subresource\_range)  
*Insert image memory barrier.*
- [VkSurfaceFormatKHR lava::find\\_surface\\_format](#) (VkPhysicalDevice [device](#), [VkSurfaceKHR](#) surface, [surface\\_format\\_request](#) request={})  
*Find the surface format.*
- bool [lava::support\\_blit](#) (VkPhysicalDevice [device](#), VkFormat format)  
*Check if format supports blitting.*
- bool [lava::support\\_vertex\\_buffer\\_format](#) (VkPhysicalDevice [device](#), VkFormat format)  
*Check if vertex buffer format is supported.*

## 5.126.1 Detailed Description

Vulkan format.

Authors

Lava Block OÜ and contributors

Copyright

Copyright (c) 2018-present, MIT License

## 5.126.2 Function Documentation

### 5.126.2.1 find\_supported\_depth\_format()

```
VkFormat_optional lava::find_supported_depth_format (
    VkPhysicalDevice physical_device)
```

Find the supported depth format.

## Parameters

<i>physical_device</i>	Physical device
------------------------	-----------------

## Returns

VkFormat\_optional Optional format

**5.126.2.2 find\_supported\_format()**

```
VkFormat_optional lava::find_supported_format (  
    VkPhysicalDevice physical_device,  
    VkFormats const & possible_formats,  
    VkImageUsageFlags usage)
```

Find the supported format.

## Parameters

<i>physical_device</i>	Physical device
<i>possible_formats</i>	List of possible formats
<i>usage</i>	Image usage flags

## Returns

VkFormat\_optional Optional format

**5.126.2.3 find\_surface\_format()**

```
VkSurfaceFormatKHR lava::find_surface_format (  
    VkPhysicalDevice device,  
    VkSurfaceKHR surface,  
    surface_format_request request = {})
```

Find the surface format.

## Parameters

<i>device</i>	Vulkan device
<i>surface</i>	Vulkan surface
<i>request</i>	Surface format request

## Returns

VkSurfaceFormatKHR Chosen surface format

**5.126.2.4 format\_align\_dim()**

```
void lava::format_align_dim (  
    VkFormat format,  
    ui32 & width,  
    ui32 & height)
```

Get align dimension of format.

## Parameters

<i>format</i>	Target format
<i>width</i>	Align width
<i>height</i>	Align height

**5.126.2.5 format\_aspect\_mask()**

```
VkImageAspectFlags lava::format_aspect_mask (  
    VkFormat format)
```

Get image aspect mask of format.

## Parameters

<i>format</i>	Target format
---------------	---------------

## Returns

VkImageAspectFlags Image aspect flags

**5.126.2.6 format\_bgr()**

```
bool lava::format_bgr (  
    VkFormat format)
```

Check if format has BGR order.

## Parameters

<i>format</i>	Format to check
---------------	-----------------

## Returns

Format has BGR order or not

**5.126.2.7 format\_block\_dim()**

```
void lava::format_block_dim (  
    VkFormat format,  
    ui32 & width,  
    ui32 & height)
```

Get block dimension of format.

## Parameters

<i>format</i>	Target format
<i>width</i>	Block width
<i>height</i>	Block height

**5.126.2.8 format\_block\_size()** [1/2]

```
ui32 lava::format_block_size (  
    VkFormat format) [inline]
```

Get format block size (with respective aspect mask)

## Parameters

<i>format</i>	Target format
---------------	---------------

## Returns

ui32 Size of block

**5.126.2.9 format\_block\_size()** [2/2]

```
ui32 lava::format_block_size (  
    VkFormat format,  
    VkImageAspectFlags aspect)
```

Get format block size.

## Parameters

<i>format</i>	Target format
<i>aspect</i>	Target aspect

## Returns

ui32 Size of block

**5.126.2.10 format\_depth()**

```
bool lava::format_depth (  
    VkFormat format)
```

Check if format is depth compatible.

## Parameters

<i>format</i>	Format to check
---------------	-----------------

## Returns

Format is depth compatible or not

**5.126.2.11 format\_depth\_stencil()**

```
bool lava::format_depth_stencil (  
    VkFormat format)
```

Check if format is depth or stencil compatible.

## Parameters

<i>format</i>	Format to check
---------------	-----------------

## Returns

Format is depth or stencil compatible or not

**5.126.2.12 format\_num\_blocks()**

```
void lava::format_num_blocks (  
    VkFormat format,  
    ui32 & width,  
    ui32 & height)
```

Get format number of blocks.

## Parameters

<i>format</i>	Target format
<i>width</i>	Number blocks width
<i>height</i>	Number blocks height

**5.126.2.13 format\_srgb()**

```
bool lava::format_srgb (  
    VkFormat format)
```

Check if format is sRGB compatible.

## Parameters

<i>format</i>	Format to check
---------------	-----------------

## Returns

Format is sRGB compatible or not

**5.126.2.14 format\_stencil()**

```
bool lava::format_stencil (
    VkFormat format)
```

Check if format is stencil compatible.

## Parameters

<i>format</i>	Format to check
---------------	-----------------

## Returns

Format is stencil compatible or not

**5.126.2.15 image\_memory\_barrier()**

```
VkImageMemoryBarrier lava::image_memory_barrier (
    VkImage image,
    VkImageLayout old_layout,
    VkImageLayout new_layout)
```

Get image memory barrier.

## Parameters

<i>image</i>	Target image
<i>old_layout</i>	Old image layout
<i>new_layout</i>	New image layout

## Returns

VkImageMemoryBarrier Image memory barrier

**5.126.2.16 insert\_image\_memory\_barrier()**

```
void lava::insert_image_memory_barrier (
    device::ptr device,
    VkCommandBuffer cmd_buffer,
    VkImage image,
    VkAccessFlags src_access_mask,
    VkAccessFlags dst_access_mask,
    VkImageLayout old_image_layout,
    VkImageLayout new_image_layout,
    VkPipelineStageFlags src_stage_mask,
    VkPipelineStageFlags dst_stage_mask,
    VkImageSubresourceRange subresource_range)
```

Insert image memory barrier.



## Parameters

<i>device</i>	Vulkan device
<i>cmd_buffer</i>	Command buffer
<i>image</i>	Target image
<i>src_access_mask</i>	Source access mask
<i>dst_access_mask</i>	Destination access mask
<i>old_image_layout</i>	Old image layout
<i>new_image_layout</i>	New image layout
<i>src_stage_mask</i>	Source pipeline stage flags
<i>dst_stage_mask</i>	Destination pipeline stage flags
<i>subresource_range</i>	Image subresource range

5.126.2.17 `set_image_layout()` [1/2]

```
void lava::set_image_layout (
    device::ptr device,
    VkCommandBuffer cmd_buffer,
    VkImage image,
    VkImageAspectFlags aspect_mask,
    VkImageLayout old_image_layout,
    VkImageLayout new_image_layout,
    VkPipelineStageFlags src_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT,
    VkPipelineStageFlags dst_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT)
```

Set the image layout.

## Parameters

<i>device</i>	Vulkan device
<i>cmd_buffer</i>	Command buffer
<i>image</i>	Target image
<i>aspect_mask</i>	Image aspect flags
<i>old_image_layout</i>	Old image layout
<i>new_image_layout</i>	New image layout
<i>src_stage_mask</i>	Source pipeline stage flags
<i>dst_stage_mask</i>	Destination pipeline stage flags

5.126.2.18 `set_image_layout()` [2/2]

```
void lava::set_image_layout (
    device::ptr device,
    VkCommandBuffer cmd_buffer,
    VkImage image,
    VkImageLayout old_image_layout,
    VkImageLayout new_image_layout,
    VkImageSubresourceRange subresource_range,
    VkPipelineStageFlags src_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT,
    VkPipelineStageFlags dst_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT)
```

Set the image layout.

## Parameters

<i>device</i>	Vulkan device
<i>cmd_buffer</i>	Command buffer
<i>image</i>	Target image
<i>old_image_layout</i>	Old image layout
<i>new_image_layout</i>	New image layout
<i>subresource_range</i>	Image subresource range
<i>src_stage_mask</i>	Source pipeline stage flags
<i>dst_stage_mask</i>	Destination pipeline stage flags

**5.126.2.19 support\_blit()**

```
bool lava::support_blit (
    VkPhysicalDevice device,
    VkFormat format)
```

Check if format supports blitting.

## Parameters

<i>device</i>	Vulkan physical device
<i>format</i>	Format to check

## Returns

Blitting is supported or not

**5.126.2.20 support\_vertex\_buffer\_format()**

```
bool lava::support_vertex_buffer_format (
    VkPhysicalDevice device,
    VkFormat format)
```

Check if vertex buffer format is supported.

## Parameters

<i>device</i>	Vulkan physical device
<i>format</i>	Format to check

## Returns

Format is supported or not

## 5.127 format.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011 #include <optional>
00012
00013 namespace lava {
00014
00016 using VkFormat_optional = std::optional<VkFormat>;
00017
00023 bool format_depth(VkFormat format);
00024
00030 bool format_stencil(VkFormat format);
00031
00037 bool format_depth_stencil(VkFormat format);
00038
00044 bool format_srgb(VkFormat format);
00045
00051 bool format_bgr(VkFormat format);
00052
00058 VkImageAspectFlags format_aspect_mask(VkFormat format);
00059
00066 void format_block_dim(VkFormat format,
00067                        ui32& width,
00068                        ui32& height);
00069
00076 void format_align_dim(VkFormat format,
00077                        ui32& width,
00078                        ui32& height);
00079
00086 void format_num_blocks(VkFormat format,
00087                          ui32& width,
00088                          ui32& height);
00089
00096 ui32 format_block_size(VkFormat format,
00097                         VkImageAspectFlags aspect);
00098
00104 inline ui32 format_block_size(VkFormat format) {
00105     return format_block_size(format, format_aspect_mask(format));
00106 };
00107
00113 VkFormat_optional find_supported_depth_format(VkPhysicalDevice physical_device);
00114
00122 VkFormat_optional find_supported_format(VkPhysicalDevice physical_device,
00123                                         VkFormats const& possible_formats,
00124                                         VkImageUsageFlags usage);
00125
00133 VkImageMemoryBarrier image_memory_barrier(VkImage image,
00134                                             VkImageLayout old_layout,
00135                                             VkImageLayout new_layout);
00136
00148 void set_image_layout(device::ptr device,
00149                       VkCommandBuffer cmd_buffer,
00150                       VkImage image,
00151                       VkImageLayout old_image_layout,
00152                       VkImageLayout new_image_layout,
00153                       VkImageSubresourceRange subresource_range,
00154                       VkPipelineStageFlags src_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT,
00155                       VkPipelineStageFlags dst_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT);
00156
00168 void set_image_layout(device::ptr device,
00169                       VkCommandBuffer cmd_buffer,
00170                       VkImage image,
00171                       VkImageAspectFlags aspect_mask,
00172                       VkImageLayout old_image_layout,
00173                       VkImageLayout new_image_layout,
00174                       VkPipelineStageFlags src_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT,
00175                       VkPipelineStageFlags dst_stage_mask = VK_PIPELINE_STAGE_ALL_COMMANDS_BIT);
00176
00190 void insert_image_memory_barrier(device::ptr device,
00191                                  VkCommandBuffer cmd_buffer,
00192                                  VkImage image,
00193                                  VkAccessFlags src_access_mask,
00194                                  VkAccessFlags dst_access_mask,
00195                                  VkImageLayout old_image_layout,
00196                                  VkImageLayout new_image_layout,
00197                                  VkPipelineStageFlags src_stage_mask,
00198                                  VkPipelineStageFlags dst_stage_mask,
00199                                  VkImageSubresourceRange subresource_range);
00200
00204 struct surface_format_request {
00206     VkFormats formats = {

```

```

00207         VK_FORMAT_B8G8R8A8_UNORM,
00208         VK_FORMAT_R8G8B8A8_UNORM,
00209         VK_FORMAT_B8G8R8_UNORM,
00210         VK_FORMAT_R8G8B8_UNORM,
00211         VK_FORMAT_B8G8R8A8_SRGB,
00212         VK_FORMAT_R8G8B8A8_SRGB,
00213         VK_FORMAT_B8G8R8_SRGB,
00214         VK_FORMAT_R8G8B8_SRGB,
00215     };
00216
00217     VkColorSpaceKHR color_space = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
00218 };
00219 };
00220
00221 VkSurfaceFormatKHR find_surface_format(VkPhysicalDevice device,
00222                                       VkSurfaceKHR surface,
00223                                       surface_format_request request = {});
00224
00225 bool support_blit(VkPhysicalDevice device,
00226                  VkFormat format);
00227
00228 bool support_vertex_buffer_format(VkPhysicalDevice device,
00229                                   VkFormat format);
00230
00231 } // namespace lava

```

## 5.128 liblava/resource/image.hpp File Reference

Vulkan image.

```
#include "liblava/base/device.hpp"
```

### Classes

- struct [lava::image\\_data](#)  
*Image data.*
- struct [lava::image](#)  
*Image.*

### Functions

- [image::s\\_ptr lava::create\\_image](#) (device::ptr device, VkFormat format, [uv2](#) size, VkImage vk\_image=0)  
*Create a new image.*
- [image::s\\_ptr lava::grab\\_image](#) (image::s\_ptr source)  
*Grab an image (with blit/copy)*

### 5.128.1 Detailed Description

Vulkan image.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.128.2 Function Documentation

### 5.128.2.1 create\_image()

```
image::s_ptr lava::create_image (
    device::ptr device,
    VkFormat format,
    uv2 size,
    VkImage vk_image = 0)
```

Create a new image.

#### Parameters

<i>device</i>	Vulkan device
<i>format</i>	Image format
<i>size</i>	Image size
<i>vk_image</i>	Vulkan image

#### Returns

image::s\_ptr Shared pointer to image

### 5.128.2.2 grab\_image()

```
image::s_ptr lava::grab_image (
    image::s_ptr source)
```

Grab an image (with blit/copy)

#### Parameters

<i>source</i>	Source image
---------------	--------------

#### Returns

image::s\_ptr Grabbed image

## 5.129 image.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/base/device.hpp"
00011
00012 namespace lava {
00013
00017 struct image_data {
00019     using s_ptr = std::shared_ptr<image_data>;
00020
00022     uv2 dimensions = uv2(0, 0);
00023
00025     ui32 channels = 0;
```

```

00026
00031     bool ready() const {
00032         return m_data != nullptr;
00033     }
00034
00039     data::ptr get_data() {
00040         return m_data;
00041     }
00042
00047     void set_data(data::ptr data) {
00048         m_data = data;
00049     }
00050
00055     size_t size() const {
00056         return channels * dimensions.x * dimensions.y;
00057     }
00058
00062     ~image_data();
00063
00064 private:
00066     data::ptr m_data = nullptr;
00067 };
00068
00072 struct image : entity {
00073     using s_ptr = std::shared_ptr<image>;
00074
00077     using s_map = std::map<id, s_ptr>;
00078
00080     using s_list = std::vector<s_ptr>;
00081
00088     static s_ptr make(VkFormat format,
00089                       VkImage vk_image = 0) {
00090         return std::make_shared<image>(format, vk_image);
00091     }
00092
00098     explicit image(VkFormat format,
00099                   VkImage vk_image = 0);
00100
00109     bool create(device::ptr device,
00110                uv2 size,
00111                VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_GPU_ONLY,
00112                VmaAllocationCreateFlags allocation_flags = 0);
00113
00118     void destroy(bool view_only = false);
00119
00123     void destroy_view() {
00124         destroy(true);
00125     }
00126
00131     device::ptr get_device() {
00132         return m_device;
00133     }
00134
00139     VkFormat get_format() const {
00140         return m_info.format;
00141     }
00142
00147     uv2 get_size() const {
00148         return {m_info.extent.width, m_info.extent.height};
00149     }
00150
00155     ui32 get_depth() const {
00156         return m_info.extent.depth;
00157     }
00158
00163     VkImage get() const {
00164         return m_vk_image;
00165     }
00166
00171     VkImageView get_view() const {
00172         return m_view;
00173     }
00174
00179     VkImageCreateInfo const& get_info() const {
00180         return m_info;
00181     }
00182
00187     VkImageViewCreateInfo const& get_view_info() const {
00188         return m_view_info;
00189     }
00190
00195     VkImageSubresourceRange const& get_subresource_range() const {
00196         return m_subresource_range;
00197     }
00198
00203     void set_flags(VkImageCreateFlagBits flags) {
00204         m_info.flags = flags;

```

```

00205     }
00206
00211     void set_tiling(VkImageTiling tiling) {
00212         m_info.tiling = tiling;
00213     }
00214
00219     void set_usage(VkImageUsageFlags usage) {
00220         m_info.usage = usage;
00221     }
00222
00227     void set_layout(VkImageLayout initial) {
00228         m_info.initialLayout = initial;
00229     }
00230
00235     void set_aspect_mask(VkImageAspectFlags aspectMask) {
00236         m_subresource_range.aspectMask = aspectMask;
00237     }
00238
00243     void set_level_count(ui32 levels) {
00244         m_subresource_range.levelCount = levels;
00245         m_info.mipLevels = levels;
00246     }
00247
00252     void set_layer_count(ui32 layers) {
00253         m_subresource_range.layerCount = layers;
00254         m_info.arrayLayers = layers;
00255     }
00256
00261     void set_component(VkComponentMapping mapping = {}) {
00262         m_view_info.components = mapping;
00263     }
00264
00269     void set_view_type(VkImageViewType type) {
00270         m_view_info.viewType = type;
00271     }
00272
00277     VmaAllocation const& get_allocation() const {
00278         return m_allocation;
00279     }
00280
00281 private:
00283     device::ptr m_device = nullptr;
00284
00286     VkImage m_vk_image = VK_NULL_HANDLE;
00287
00289     VkImageCreateInfo m_info;
00290
00292     VmaAllocation m_allocation = nullptr;
00293
00295     VkImageView m_view = VK_NULL_HANDLE;
00296
00298     VkImageViewCreateInfo m_view_info;
00299
00301     VkImageSubresourceRange m_subresource_range;
00302 };
00303
00312 image::s_ptr create_image(device::ptr device,
00313                          VkFormat format,
00314                          uv2 size,
00315                          VkImage vk_image = 0);
00316
00322 image::s_ptr grab_image(image::s_ptr source);
00323
00324 } // namespace lava

```

## 5.130 liblava/resource/mesh.hpp File Reference

Vulkan mesh.

```

#include "liblava/core/misc.hpp"
#include "liblava/resource/buffer.hpp"
#include "liblava/resource/primitive.hpp"
#include "liblava/util/hex.hpp"
#include "liblava/util/log.hpp"

```

## Classes

- struct [lava::mesh\\_template\\_data< T >](#)  
*Templated mesh data.*
- struct [lava::mesh\\_template< T >](#)  
*Temporary templated mesh.*
- struct [lava::mesh\\_meta](#)  
*Mesh meta.*

## Typedefs

- using [lava::mesh\\_data](#) = [mesh\\_template\\_data<vertex>](#)  
*Mesh data with default vertex.*
- using [lava::mesh](#) = [mesh\\_template<vertex>](#)  
*Mesh with default vertex.*
- using [lava::mesh\\_registry](#) = [id\\_registry<mesh, mesh\\_meta>](#)  
*Mesh registry.*

## Functions

- [template<typename T = vertex, bool generate\\_colors = true, bool generate\\_normals = true, bool generate\\_uvs = true, bool has\\_colors = true, bool has\\_normals = true, bool has\\_uvs = true>](#)  
[mesh\\_template\\_data< T >](#) [lava::create\\_mesh\\_data](#) ([mesh\\_type](#) type)  
*Create a new primitive mesh\_data.*
- [template<typename T = vertex, bool generate\\_colors = true, bool generate\\_normals = true, bool generate\\_uvs = true, bool has\\_colors = true, bool has\\_normals = true, bool has\\_uvs = true>](#)  
[std::shared\\_ptr< mesh\\_template< T > >](#) [lava::create\\_mesh](#) ([device::ptr](#) &device, [mesh\\_type](#) type)  
*Create a new primitive mesh.*
- [template<typename PosType, size\\_t vert\\_count, bool is\\_complex>](#)  
[constexpr std::array< PosType, vert\\_count >](#) [lava::make\\_primitive\\_positions\\_cube](#) ()  
*Make primitive positions for cube.*
- [template<bool is\\_complex>](#)  
[std::vector< index >](#) [lava::make\\_primitive\\_indices\\_cube](#) ()  
*Make primitive indices for cube.*
- [template<typename NormType >](#)  
[constexpr std::array< NormType, 6 >](#) [lava::make\\_primitive\\_normals\\_cube](#) ()  
*Make primitive normals for cube.*
- [template<typename UVType >](#)  
[constexpr std::array< UVType, 24 >](#) [lava::make\\_primitive\\_uvs\\_cube](#) ()  
*Make primitive uvs for cube.*

### 5.130.1 Detailed Description

Vulkan mesh.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License



## 5.130.2 Function Documentation

### 5.130.2.1 create\_mesh()

```
template<typename T = vertex, bool generate_colors = true, bool generate_normals = true, bool
generate_uvcs = true, bool has_colors = true, bool has_normals = true, bool has_uvcs = true>
std::shared_ptr< mesh_template< T > > lava::create_mesh (
    device::ptr & device,
    mesh_type type)
```

Create a new primitive mesh.

#### Template Parameters

<i>T</i>	Type of vertex struct
<i>generate_colors</i>	If color may be generated
<i>generate_normals</i>	If normals may be generated
<i>generate_uvcs</i>	If UVs may be generated
<i>has_colors</i>	On MSVC, specifies if a <code>color</code> field exists
<i>has_normals</i>	On MSVC, specifies if a <code>normal</code> field exists
<i>has_uvcs</i>	On MSVC, specifies if a <code>uv</code> field exists

#### Parameters

<i>device</i>	Vulkan device
<i>type</i>	Mesh type

#### Returns

`std::shared_ptr<mesh_template<T>>` Shared pointer to mesh

### 5.130.2.2 create\_mesh\_data()

```
template<typename T = vertex, bool generate_colors = true, bool generate_normals = true, bool
generate_uvcs = true, bool has_colors = true, bool has_normals = true, bool has_uvcs = true>
mesh_template_data< T > lava::create_mesh_data (
    mesh_type type)
```

Create a new primitive mesh\_data.

#### Template Parameters

<i>T</i>	Type of vertex struct
<i>generate_colors</i>	If color may be generated
<i>generate_normals</i>	If normals may be generated
<i>generate_uvcs</i>	If UVs may be generated
<i>has_colors</i>	On MSVC, specifies if a <code>color</code> field exists
<i>has_normals</i>	On MSVC, specifies if a <code>normal</code> field exists
<i>has_uvcs</i>	On MSVC, specifies if a <code>uv</code> field exists

**Parameters**

<i>type</i>	Mesh type
-------------	-----------

**Returns**

mesh\_template\_data<T> Mesh data

**5.130.2.3 make\_primitive\_indices\_cube()**

```
template<bool is_complex>
std::vector< index > lava::make_primitive_indices_cube ()
```

Make primitive indices for cube.

**Template Parameters**

<i>is_complex</i>	Complex state
-------------------	---------------

**Returns**

std::vector<index> Array for indices

**5.130.2.4 make\_primitive\_normals\_cube()**

```
template<typename NormType >
std::array< NormType, 6 > lava::make_primitive_normals_cube () [constexpr]
```

Make primitive normals for cube.

**Template Parameters**

<i>NormType</i>	Type of normal
-----------------	----------------

**Returns**

constexpr std::array<NormType, 6> Array of normals

**5.130.2.5 make\_primitive\_positions\_cube()**

```
template<typename PosType , size_t vert_count, bool is_complex>
std::array< PosType, vert_count > lava::make_primitive_positions_cube () [constexpr]
```

Make primitive positions for cube.

## Template Parameters

<i>PosType</i>	Type of position
<i>vert_count</i>	Number of vertices
<i>is_complex</i>	Complex state

## Returns

constexpr std::array<PosType, vert\_count> Array of positions

## 5.130.2.6 make\_primitive\_uvs\_cube()

```
template<typename UVType >
std::array< UVType, 24 > lava::make_primitive_uvs_cube () [constexpr]
```

Make primitive uvs for cube.

## Template Parameters

<i>UVType</i>	Type of uv
---------------	------------

## Returns

constexpr std::array<UVType, 24> Array of uvs

## 5.131 mesh.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/misc.hpp"
00011 #include "liblava/resource/buffer.hpp"
00012 #include "liblava/resource/primitive.hpp"
00013 #include "liblava/util/hex.hpp"
00014 #include "liblava/util/log.hpp"
00015
00016 namespace lava {
00017
00022 template <typename T = vertex>
00023 struct mesh_template_data {
00025     std::vector<T> vertices;
00026
00028     index_list indices;
00029
00035     template <typename PosType = r32>
00036     void move(std::array<PosType, 3> offset) {
00037         for (T& vertex : vertices) {
00038             for (auto i = 0u; i < 3; ++i) {
00039                 vertex.position[i] += offset[i];
00040             }
00041         }
00042     }
00043
00048     void scale(auto factor) {
00049         for (T& vertex : vertices) {
00050             for (auto i = 0u; i < 3; ++i) {
00051                 vertex.position[i] *= factor;
00052             }
00053         }
00054     }
00055 }
```

```

00061     template <typename PosType = r32>
00062     void scale_vector(std::array<PosType, 3> factors) {
00063         for (T& vertex : vertices) {
00064             for (auto i = 0u; i < 3; ++i) {
00065                 vertex.position[i] *= factors[i];
00066             }
00067         }
00068     }
00069 };
00070
00075 template <typename T = vertex>
00076 struct mesh_template : entity {
00077     using s_ptr = std::shared_ptr<mesh_template<T>;
00078
00081     using s_map = std::map<id, s_ptr>;
00082
00084     using s_list = std::vector<s_ptr>;
00085
00087     using vertex_list = std::vector<T>;
00088
00093     static s_ptr make() {
00094         return std::make_shared<mesh_template<T>>();
00095     }
00096
00100     ~mesh_template() {
00101         destroy();
00102     }
00103
00111     bool create(device::ptr device,
00112                bool mapped = false,
00113                VmaMemoryUsage memory_usage = VMA_MEMORY_USAGE_CPU_TO_GPU);
00114
00118     void destroy();
00119
00124     void bind(VkCommandBuffer cmd_buf) const;
00125
00130     void draw(VkCommandBuffer cmd_buf) const;
00131
00136     void bind_draw(VkCommandBuffer cmd_buf) const {
00137         bind(cmd_buf);
00138         draw(cmd_buf);
00139     }
00140
00145     bool empty() const {
00146         return m_data.vertices.empty();
00147     }
00148
00153     void set_data(mesh_template_data<T> const& value) {
00154         m_data = value;
00155     }
00156
00161     mesh_template_data<T>& get_data() {
00162         return m_data;
00163     }
00164
00169     void add_data(mesh_template_data<T> const& value) {
00170         m_data = value;
00171     }
00172
00177     vertex_list& get_vertices() {
00178         return m_data.vertices;
00179     }
00180
00185     vertex_list const& get_vertices() const {
00186         return m_data.vertices;
00187     }
00188
00193     ui32 get_vertices_count() const {
00194         return to_ui32(m_data.vertices.size());
00195     }
00196
00201     index_list& get_indices() {
00202         return m_data.indices;
00203     }
00204
00209     index_list const& get_indices() const {
00210         return m_data.indices;
00211     }
00212
00217     ui32 get_indices_count() const {
00218         return to_ui32(m_data.indices.size());
00219     }
00220
00225     bool reload();
00226
00231     buffer::s_ptr get_vertex_buffer() {
00232         return m_vertex_buffer;

```

```

00233     }
00234
00239     buffer::s_ptr get_index_buffer() {
00240         return m_index_buffer;
00241     }
00242
00243 private:
00244     device::ptr m_device = nullptr;
00245
00246     mesh_template_data<T> m_data;
00247
00248     buffer::s_ptr m_vertex_buffer;
00249
00250     buffer::s_ptr m_index_buffer;
00251
00252     bool m_mapped = false;
00253
00254     VmaMemoryUsage m_memory_usage = VMA_MEMORY_USAGE_CPU_TO_GPU;
00255 };
00256
00257 //-----
00258 template <typename T>
00259 void mesh_template<T>::bind(VkCommandBuffer cmd_buf) const {
00260     if (m_vertex_buffer && m_vertex_buffer->valid()) {
00261         std::array<VkDeviceSize, 1> const buffer_offsets = {0};
00262         std::array<VkBuffer, 1> const buffers = {m_vertex_buffer->get()};
00263
00264         vkCmdBindVertexBuffers(cmd_buf, 0,
00265                                to_ui32(buffers.size()), buffers.data(),
00266                                buffer_offsets.data());
00267     }
00268
00269     if (m_index_buffer && m_index_buffer->valid())
00270         vkCmdBindIndexBuffer(cmd_buf,
00271                               m_index_buffer->get(),
00272                               0,
00273                               VK_INDEX_TYPE_UINT32);
00274 }
00275
00276 //-----
00277 template <typename T>
00278 void mesh_template<T>::draw(VkCommandBuffer cmd_buf) const {
00279     if (!m_data.indices.empty())
00280         vkCmdDrawIndexed(cmd_buf,
00281                           to_ui32(m_data.indices.size()),
00282                           1, 0, 0, 0);
00283     else
00284         vkCmdDraw(cmd_buf,
00285                   to_ui32(m_data.vertices.size()),
00286                   1, 0, 0);
00287 }
00288
00289 //-----
00290 template <typename T>
00291 void mesh_template<T>::destroy() {
00292     m_vertex_buffer = nullptr;
00293     m_index_buffer = nullptr;
00294     m_device = nullptr;
00295 }
00296
00297 //-----
00298 template <typename T>
00299 bool mesh_template<T>::reload() {
00300     auto dev = m_device;
00301     destroy();
00302
00303     return create(dev, m_mapped, m_memory_usage);
00304 }
00305
00306 //-----
00307 template <typename T = vertex,
00308           bool generate_colors = true,
00309           bool generate_normals = true,
00310           bool generate_uvs = true,
00311           bool has_colors = true,
00312           bool has_normals = true,
00313           bool has_uvs = true>
00314 mesh_template_data<T> create_mesh_data(mesh_type type);
00315
00316 template <typename T = vertex,
00317           bool generate_colors = true,
00318           bool generate_normals = true,
00319           bool generate_uvs = true,
00320           bool has_colors = true,
00321           bool has_normals = true,
00322           bool has_uvs = true>
00323 std::shared_ptr<mesh_template<T>> create_mesh(device::ptr& device,
00324                                                mesh_type type);

```

```

00355
00356 //-----
00357 template <typename T>
00358 bool mesh_template<T>::create(device::ptr dev,
00359                             bool m,
00360                             VmaMemoryUsage mu) {
00361     m_device = dev;
00362     m_mapped = m;
00363     m_memory_usage = mu;
00364
00365     if (!m_data.vertices.empty()) {
00366         m_vertex_buffer = buffer::make();
00367
00368         if (!m_vertex_buffer->create(m_device,
00369                                     m_data.vertices.data(),
00370                                     sizeof(T) * m_data.vertices.size(),
00371                                     VK_BUFFER_USAGE_VERTEX_BUFFER_BIT,
00372                                     m_mapped,
00373                                     m_memory_usage)) {
00374             logger()->error("create mesh vertex buffer");
00375             return false;
00376         }
00377     }
00378
00379     if (!m_data.indices.empty()) {
00380         m_index_buffer = buffer::make();
00381
00382         if (!m_index_buffer->create(m_device,
00383                                     m_data.indices.data(),
00384                                     sizeof(ui32) * m_data.indices.size(),
00385                                     VK_BUFFER_USAGE_INDEX_BUFFER_BIT,
00386                                     m_mapped,
00387                                     m_memory_usage)) {
00388             logger()->error("create mesh index buffer");
00389             return false;
00390         }
00391     }
00392
00393     return true;
00394 }
00395
00403 template <typename PostType, size_t vert_count, bool is_complex>
00404 constexpr std::array<PostType, vert_count> make_primitive_positions_cube();
00405
00406 //-----
00407 // NOTE: The C++20 spec allows std::vector<T> to be constexpr
00408 // g++ does not currently implement this feature, however
00409 //-----
00410
00416 template <bool is_complex>
00417 std::vector<index> make_primitive_indices_cube();
00418
00424 template <typename NormType>
00425 constexpr std::array<NormType, 6> make_primitive_normals_cube();
00426
00432 template <typename UVType>
00433 constexpr std::array<UVType, 24> make_primitive_uvcs_cube();
00434
00435 //-----
00436 template <typename T,
00437         bool generate_colors,
00438         bool generate_normals,
00439         bool generate_uvcs,
00440         bool has_colors,
00441         bool has_normals,
00442         bool has_uvcs>
00443 std::shared_ptr<mesh_template<T>> create_mesh(device::ptr& device,
00444                                              mesh_type type) {
00445     std::shared_ptr<mesh_template<T>> return_mesh =
00446         std::make_shared<mesh_template<T>>();
00447
00448     return_mesh->add_data(create_mesh_data<T,
00449                          generate_colors,
00450                          generate_normals,
00451                          generate_uvcs,
00452                          has_colors,
00453                          has_normals,
00454                          has_uvcs>(type));
00455     return_mesh->create(device);
00456     return return_mesh;
00457 }
00458
00459 //-----
00460 template <typename T,
00461         bool generate_colors,
00462         bool generate_normals,
00463         bool generate_uvcs,

```

```

00464         bool has_colors,
00465         bool has_normals,
00466         bool has_uv>
00467 mesh_template_data<T> create_mesh_data(mesh_type type) {
00468     mesh_template_data<T> return_mesh_data;
00469
00470     constexpr bool auto_position = requires(const T t) {
00471         t.position;
00472     };
00473     static_assert(auto_position,
00474         "Vertex struct `T` must contain field `position`");
00475
00476     constexpr bool auto_colors = requires(const T t) {
00477         t.color;
00478     };
00479     constexpr bool auto_normals = requires(const T t) {
00480         t.normal;
00481     };
00482     constexpr bool auto_uv = requires(const T t) {
00483         t.uv;
00484     };
00485
00486     using PosType = decltype(T::position);
00487
00488     switch (type) {
00489     case mesh_type::cube: {
00490         return_mesh_data.indices.reserve(36);
00491         return_mesh_data.indices = make_primitive_indices_cube<(generate_normals
00492             && auto_normals)>();
00493         constexpr size_t vert_count = (generate_normals && auto_normals) ? 24 : 8;
00494         return_mesh_data.vertices.reserve(vert_count);
00495         auto positions = make_primitive_positions_cube<PosType, vert_count,
00496             (generate_normals && auto_normals)>();
00497         for (size_t i = 0; i < vert_count; i++) {
00498             T vert;
00499             vert.position = positions[i];
00500             if constexpr (generate_normals && auto_normals) {
00501                 // this array is generated inside of every loop because
00502                 // that makes the scoping rules simplest to follow
00503                 // my expectation is that a compiler should be able
00504                 // to trivially optimize this
00505                 using NormType = decltype(T::normal);
00506                 auto normals = make_primitive_normals_cube<NormType>();
00507                 vert.normal = normals[i / 4];
00508             }
00509             if constexpr (generate_uv && auto_uv) {
00510                 using UVType = decltype(T::uv);
00511                 auto uvs = make_primitive_uvs_cube<UVType>();
00512                 vert.uv = uvs[i];
00513             }
00514             return_mesh_data.vertices.push_back(vert);
00515         }
00516         break;
00517     }
00518
00519     case mesh_type::triangle: {
00520         return_mesh_data.vertices.reserve(3);
00521         T vert_one;
00522         vert_one.position = {1, 1, 0};
00523         T vert_two;
00524         vert_two.position = {-1, 1, 0};
00525         T vert_three;
00526         vert_three.position = {0, -1, 0};
00527         if constexpr (generate_uv && auto_uv) {
00528             vert_one.uv = {1, 1};
00529             vert_two.uv = {0, 1};
00530             vert_three.uv = {0.5, 0};
00531         }
00532         if constexpr (generate_normals && auto_normals) {
00533             vert_one.normal = {1, 1, 0};
00534             vert_two.normal = {-1, 1, 0};
00535             vert_three.normal = {0, -1, 0};
00536         }
00537         return_mesh_data.vertices.push_back(vert_one);
00538         return_mesh_data.vertices.push_back(vert_two);
00539         return_mesh_data.vertices.push_back(vert_three);
00540         break;
00541     }
00542
00543     case mesh_type::quad: {
00544         return_mesh_data.vertices.reserve(4);
00545         return_mesh_data.indices.reserve(6);
00546         T vert_one;
00547         vert_one.position = {1, 1, 0};
00548         T vert_two;
00549         vert_two.position = {-1, 1, 0};
00550         T vert_three;

```

```

00551     vert_three.position = {-1, -1, 0};
00552     T vert_four;
00553     vert_four.position = {1, -1, 0};
00554     if constexpr (generate_uvcs && auto_uvcs) {
00555         vert_one.uv = {1, 1};
00556         vert_two.uv = {0, 1};
00557         vert_three.uv = {0, 0};
00558         vert_four.uv = {1, 0};
00559     }
00560     if constexpr (generate_normals && auto_normals) {
00561         vert_one.normal = {0, 0, 1};
00562         vert_two.normal = {0, 0, 1};
00563         vert_three.normal = {0, 0, 1};
00564         vert_four.normal = {0, 0, 1};
00565     }
00566     // clang-format off
00567     return_mesh_data.indices = {
00568         0, 1, 2,
00569         2, 3, 0,
00570     };
00571     // clang-format on
00572     return_mesh_data.vertices.push_back(vert_one);
00573     return_mesh_data.vertices.push_back(vert_two);
00574     return_mesh_data.vertices.push_back(vert_three);
00575     return_mesh_data.vertices.push_back(vert_four);
00576     break;
00577 }
00578
00579 case mesh_type::hexagon: {
00580     return_mesh_data.vertices.reserve(7);
00581     return_mesh_data.indices.reserve(18);
00582     hex_layout layout;
00583     layout.orientation = hex_layout_point_y;
00584     layout.size = {1, 1};
00585     auto hex_corners = hex_polygon_corners(layout, {});
00586     T vert_center;
00587     vert_center.position = {0, 0, 0};
00588     T vert_sw;
00589     PosType temp;
00590     temp[0] = hex_corners.at(0).x;
00591     temp[1] = hex_corners.at(0).y;
00592     temp[2] = 0;
00593     vert_sw.position = temp;
00594     T vert_nw;
00595     temp[0] = hex_corners.at(1).x;
00596     temp[1] = hex_corners.at(1).y;
00597     vert_nw.position = temp;
00598     T vert_n;
00599     temp[0] = hex_corners.at(2).x;
00600     temp[1] = hex_corners.at(2).y;
00601     vert_n.position = temp;
00602     T vert_ne;
00603     temp[0] = hex_corners.at(3).x;
00604     temp[1] = hex_corners.at(3).y;
00605     vert_ne.position = temp;
00606     T vert_se;
00607     temp[0] = hex_corners.at(4).x;
00608     temp[1] = hex_corners.at(4).y;
00609     vert_se.position = temp;
00610     T vert_s;
00611     temp[0] = hex_corners.at(5).x;
00612     temp[1] = hex_corners.at(5).y;
00613     vert_s.position = temp;
00614     if constexpr (generate_uvcs && auto_uvcs) {
00615         vert_center.uv = {0, 0};
00616         vert_sw.uv = {1, 0};
00617         vert_nw.uv = {0, 1};
00618         vert_n.uv = {1, 1};
00619         vert_ne.uv = {1, 0};
00620         vert_se.uv = {0, 1};
00621         vert_s.uv = {1, 1};
00622     }
00623     if constexpr (generate_normals && auto_normals) {
00624         vert_center.normal = {0, 0, 1};
00625         vert_sw.normal = {0, 0, 1};
00626         vert_nw.normal = {0, 0, 1};
00627         vert_n.normal = {0, 0, 1};
00628         vert_ne.normal = {0, 0, 1};
00629         vert_se.normal = {0, 0, 1};
00630         vert_s.normal = {0, 0, 1};
00631     }
00632     // clang-format off
00633     return_mesh_data.indices = {
00634         0, 1, 6, 0, 6, 5, 0, 5, 4, 0, 4, 3, 0, 3, 2, 0, 2, 1
00635     };
00636     // clang-format on
00637     return_mesh_data.vertices.push_back(vert_center);

```



```

00638         return_mesh_data.vertices.push_back(vert_sw);
00639         return_mesh_data.vertices.push_back(vert_nw);
00640         return_mesh_data.vertices.push_back(vert_n);
00641         return_mesh_data.vertices.push_back(vert_ne);
00642         return_mesh_data.vertices.push_back(vert_se);
00643         return_mesh_data.vertices.push_back(vert_s);
00644         break;
00645     }
00646
00647     case mesh_type::none:
00648     default:
00649         break;
00650 }
00651
00652 if constexpr (generate_colors && auto_colors) {
00653     for (auto& vert : return_mesh_data.vertices) {
00654         // this does not work on glm vectors
00655         // for (auto& this_color : vert.color) {
00656             //     for (auto& color_component : this_color) {
00657                 //         color_component = 1;
00658             //     }
00659         // }
00660         if constexpr (std::is_same_v<decltype(vert.color), glm::vec3>) {
00661             vert.color = {1, 1, 1};
00662         } else if constexpr (std::is_same_v<decltype(vert.color), glm::vec4>) {
00663             vert.color = {1, 1, 1, 1};
00664         } else {
00665             for (size_t i = 0; i < vert.color.size(); i++) {
00666                 vert.color[i] = 1;
00667             }
00668         }
00669     }
00670 }
00671
00672 return return_mesh_data;
00673 }
00674
00675 //-----
00676 template <typename PosType,
00677           size_t vert_count,
00678           bool is_complex>
00679 constexpr std::array<PosType, vert_count> make_primitive_positions_cube() {
00680     // clang-format off
00681     if constexpr (is_complex) {
00682         std::array<PosType, 24> const positions = {{
00683             // front
00684             { 1, 1, 1 }, { -1, 1, 1 }, { -1, -1, 1 }, { 1, -1, 1 },
00685             // back
00686             { 1, 1, -1 }, { -1, 1, -1 }, { -1, -1, -1 }, { 1, -1, -1 },
00687             // left
00688             { -1, 1, 1 }, { -1, 1, -1 }, { -1, -1, -1 }, { -1, -1, 1 },
00689             // right
00690             { 1, 1, 1 }, { 1, -1, 1 }, { 1, -1, -1 }, { 1, 1, -1 },
00691             // bottom
00692             { 1, 1, 1 }, { -1, 1, 1 }, { -1, 1, -1 }, { 1, 1, -1 },
00693             // top
00694             { 1, -1, 1 }, { -1, -1, 1 }, { -1, -1, -1 }, { 1, -1, -1 },
00695         }};
00696         return positions;
00697     } else {
00698         std::array<PosType, 8> const positions = {{
00699             { -1, -1, -1 },
00700             { -1, -1, 1 },
00701             { -1, 1, -1 },
00702             { -1, 1, 1 },
00703             { 1, -1, -1 },
00704             { 1, -1, 1 },
00705             { 1, 1, -1 },
00706             { 1, 1, 1 },
00707         }};
00708         return positions;
00709     }
00710     // clang-format on
00711 }
00712
00713 //-----
00714 template <bool is_complex>
00715 std::vector<index> make_primitive_indices_cube() {
00716     // clang-format off
00717     if constexpr (is_complex) {
00718         std::vector<index> const indices = {
00719             0, 1, 2,
00720             2, 3, 0,
00721             4, 7, 6,
00722             6, 5, 4,
00723             8, 9, 10,
00724             10, 11, 8,

```

```

00725         12, 13, 14,
00726         14, 15, 12,
00727         16, 19, 18,
00728         18, 17, 16,
00729         20, 21, 22,
00730         22, 23, 20,
00731     };
00732     return indices;
00733 } else {
00734     // clockwise winding order
00735     std::vector<index> const indices = {
00736         // left
00737         0, 1, 2,
00738         2, 1, 3,
00739         // right
00740         4, 5, 6,
00741         6, 5, 7,
00742         // top
00743         0, 1, 4,
00744         4, 1, 5,
00745         // bottom
00746         2, 3, 6,
00747         6, 3, 7,
00748         // back
00749         3, 1, 5,
00750         5, 7, 3,
00751         // front
00752         2, 0, 4,
00753         4, 6, 2,
00754     };
00755     return indices;
00756 }
00757 // clang-format on
00758 }
00759
00760 //-----
00761 template <typename NormType>
00762 constexpr std::array<NormType, 6> make_primitive_normals_cube() {
00763     // clang-format off
00764     // front, back, left, right, bottom, and top normals, in that order
00765     std::array<NormType, 6> const normals = {{ { 0, 0, 1 }, { 0, 0, -1 },
00766         { -1, 0, 0 }, { 1, 0, 0 },
00767         { 0, 1, 0 }, { 0, -1, 0 }, }};
00768     // clang-format on
00769     return normals;
00770 }
00771
00772 //-----
00773 template <typename UVType>
00774 constexpr std::array<UVType, 24> make_primitive_uvs_cube() {
00775     // clang-format off
00776     std::array<UVType, 24> const uvs = {{
00777         { 1, 1 }, { 0, 1 }, { 0, 0 }, { 1, 0 }, // front
00778         { 0, 1 }, { 1, 1 }, { 1, 0 }, { 0, 0 }, // back
00779         { 1, 1 }, { 0, 1 }, { 0, 0 }, { 1, 0 }, // left
00780         { 0, 1 }, { 0, 0 }, { 1, 0 }, { 1, 1 }, // right
00781         { 1, 0 }, { 0, 0 }, { 0, 1 }, { 1, 1 }, // bottom
00782         { 1, 1 }, { 0, 1 }, { 0, 0 }, { 1, 0 }, // top
00783     }};
00784     // clang-format on
00785     return uvs;
00786 }
00787
00788 using mesh_data = mesh_template_data<vertex>;
00789
00790 using mesh = mesh_template<vertex>;
00791
00792 struct mesh_meta {
00793     string filename;
00794     mesh_type type = mesh_type::none;
00795 };
00796
00797 using mesh_registry = id_registry<mesh, mesh_meta>;
00798 } // namespace lava

```

## 5.132 liblava/resource/primitive.hpp File Reference

Vulkan primitives.

```
#include "liblava/util/math.hpp"
```

## Classes

- struct [lava::vertex](#)  
*Vertex.*

## Enumerations

- enum class [lava::mesh\\_type](#) : index {  
  **none** = 0 , **cube** , **triangle** , **quad** ,  
  **hexagon** }  
*Mesh types.*

### 5.132.1 Detailed Description

Vulkan primitives.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.133 primitive.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/util/math.hpp"
00011
00012 namespace lava {
00013
00017 struct vertex {
00019     using list = std::vector<vertex>;
00020
00022     v3 position;
00023
00025     v4 color;
00026
00028     v2 uv;
00029
00031     v3 normal;
00032
00038     bool operator==(vertex const& other) const {
00039         return position == other.position
00040             && color == other.color
00041             && uv == other.uv
00042             && normal == other.normal;
00043     }
00044 };
00045
00049 enum class mesh_type : index {
00050     none = 0,
00051     cube,
00052     triangle,
00053     quad,
00054     hexagon,
00055 };
00056
00057 } // namespace lava
```

## 5.134 liblava/resource/texture.hpp File Reference

Vulkan texture.

```
#include "liblava/resource/buffer.hpp"
#include "liblava/resource/image.hpp"
```

### Classes

- struct [lava::texture\\_file](#)  
*Texture file path with format.*
- struct [lava::texture](#)  
*Texture.*
- struct [lava::texture::mip\\_level](#)  
*Texture mip level.*
- struct [lava::texture::layer](#)  
*Texture layer.*
- struct [lava::staging](#)  
*Texture staging.*

### Typedefs

- using [lava::texture\\_registry](#) = [id\\_registry](#)<[texture](#), [texture\\_file](#)>  
*Texture registry.*

### Enumerations

- enum class [lava::texture\\_type](#) : index { **none** = 0 , **tex\_2d** , **array** , **cube\_map** }  
*Texture types.*

### 5.134.1 Detailed Description

Vulkan texture.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.135 texture.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/resource/buffer.hpp"
00011 #include "liblava/resource/image.hpp"
00012
00013 namespace lava {
00014
00018 enum class texture_type : index {
00019     none = 0,
00020     tex_2d,
00021     array,
00022     cube_map
00023 };
00024
00028 struct texture_file {
00030     using list = std::vector<texture_file>;
00031
00033     string path;
00034
00036     VkFormat format = VK_FORMAT_UNDEFINED;
00037 };
00038
00042 struct texture : entity {
00044     using s_ptr = std::shared_ptr<texture>;
00045
00047     using s_map = std::map<id, s_ptr>;
00048
00050     using s_list = std::vector<s_ptr>;
00051
00055     struct mip_level {
00057         using list = std::vector<mip_level>;
00058
00060         uv2 extent{};
00061
00063         ui32 size = 0;
00064     };
00065
00069     struct layer {
00071         using list = std::vector<layer>;
00072
00074         mip_level::list levels;
00075     };
00076
00081     static s_ptr make() {
00082         return std::make_shared<texture>();
00083     }
00084
00088     ~texture() {
00089         destroy();
00090     }
00091
00101     bool create(device::ptr device,
00102                uv2 size,
00103                VkFormat format,
00104                layer::list const& layers = {},
00105                texture_type type = texture_type::tex_2d);
00106
00110     void destroy();
00111
00118     bool upload(void const* data,
00119                size_t data_size);
00120
00126     bool stage(VkCommandBuffer cmd_buffer);
00127
00131     void destroy_upload_buffer();
00132
00137     VkDescriptorImageInfo const* get_descriptor_info() const {
00138         return &m_descriptor;
00139     }
00140
00145     image::s_ptr get_image() {
00146         return m_img;
00147     }
00148
00153     uv2 get_size() const {
00154         return m_img ? m_img->get_size() : uv2();
00155     }
00156
00161     texture_type get_type() const {
00162         return m_type;
00163     }

```

```

00164
00169     VkFormat get_format() const {
00170         return m_img ? m_img->get_format() : VK_FORMAT_UNDEFINED;
00171     }
00172
00173 private:
00175     image::s_ptr m_img;
00176
00178     texture_type m_type = texture_type::none;
00179
00181     layer::list m_layers;
00182
00184     VkSampler m_sampler = 0;
00185
00187     VkDescriptorImageInfo m_descriptor = {};
00188
00190     buffer::s_ptr m_upload_buffer;
00191 };
00192
00196 struct staging {
00198     using ptr = staging*;
00199
00204     void add(texture::s_ptr texture) {
00205         m_todo.push_back(texture);
00206     }
00207
00214     bool stage(VkCommandBuffer cmd_buf,
00215               index frame);
00216
00220     void clear() {
00221         m_todo.clear();
00222         m_staged.clear();
00223     }
00224
00229     bool busy() const {
00230         return !m_todo.empty() || !m_staged.empty();
00231     }
00232
00233 private:
00235     texture::s_list m_todo;
00236
00238     using frame_stage_map = std::map<index, texture::s_list>;
00239
00241     frame_stage_map m_staged;
00242 };
00243
00245 using texture_registry = id_registry<texture, texture_file>;
00246
00247 } // namespace lava

```

## 5.136 liblava/test.hpp File Reference

Unit tests.

```

#include "catch2/catch_test_macros.hpp"
#include "liblava/lava.hpp"

```

### 5.136.1 Detailed Description

Unit tests.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.137 test.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #include "catch2/catch_test_macros.hpp"
00009 #include "liblava/lava.hpp"
00010
00011 using namespace lava;
```

## 5.138 liblava/util.hpp File Reference

Util module.

```
#include "liblava/util/hex.hpp"
#include "liblava/util/layer.hpp"
#include "liblava/util/log.hpp"
#include "liblava/util/math.hpp"
#include "liblava/util/random.hpp"
#include "liblava/util/telegram.hpp"
#include "liblava/util/thread.hpp"
```

### 5.138.1 Detailed Description

Util module.

#### Author

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

## 5.139 util.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/util/hex.hpp"
00011 #include "liblava/util/layer.hpp"
00012 #include "liblava/util/log.hpp"
00013 #include "liblava/util/math.hpp"
00014 #include "liblava/util/random.hpp"
00015 #include "liblava/util/telegram.hpp"
00016 #include "liblava/util/thread.hpp"
```

## 5.140 liblava/util/hex.hpp File Reference

Hex point, cell and grid.

```
#include "liblava/core/types.hpp"
#include <cmath>
#include <numbers>
#include <unordered_map>
```

## Classes

- struct [lava::hex\\_point](#)  
*Hex point.*
- struct [lava::hex\\_cell](#)  
*Hex cell.*
- struct [lava::hex\\_fractional\\_cell](#)  
*Hex fractional cell.*
- struct [lava::hex\\_offset\\_coord](#)  
*Hex offset coordinates.*
- struct [lava::hex\\_orientation](#)  
*Hex orientation.*
- struct [lava::hex\\_layout](#)  
*Hex layout.*
- struct [lava::hex\\_grid](#)  
*Hex grid.*

## Typedefs

- using [lava::hex\\_frac\\_cell](#) = [hex\\_fractional\\_cell](#)  
*Hex fractional cell.*
- using [lava::hex\\_doubled\\_coord](#) = [hex\\_offset\\_coord](#)  
*Hex doubled coordinates.*

## Enumerations

- enum class [lava::hex\\_offset](#) : i32 { **odd** = -1 , **even** = 1 }  
*Hex offsets.*
- enum class [lava::hex\\_cardinal\\_direction](#) : index {  
**NE** = 0 , **E** , **SE** , **SW** ,  
**W** , **NW** }  
*Hex cardinal directions.*

## Functions

- i32 [lava::hex\\_get\\_s](#) (i32 q, i32 r)  
*Get S axis from Q and R axes.*
- [hex\\_cell](#) [lava::hex\\_cell\\_from\\_pair](#) ([hex\\_cell::pair](#) const &pair)  
*Get hex cell from pair.*
- bool [lava::hex\\_is\\_valid](#) ([hex\\_cell](#) const &cell)  
*Check if hex cell is valid.*
- [hex\\_cell](#) [lava::hex\\_direction](#) (index direction)  
*Get the hex cell from direction.*
- [hex\\_cell](#) [lava::hex\\_neighbor](#) ([hex\\_cell](#) const &cell, index direction)  
*Get the neighbor of hex cell by direction.*
- [hex\\_cell](#) [lava::hex\\_diagonal](#) (index direction)  
*Get the diagonal from direction.*
- [hex\\_cell](#) [lava::hex\\_diagonal\\_neighbor](#) ([hex\\_cell](#) const &cell, index direction)  
*Get the diagonal neighbor of hex cell by direction.*



- [i32 lava::hex\\_length](#) ([hex\\_cell](#) const &cell)  
*Get the length of hex cell.*
- [i32 lava::hex\\_distance](#) ([hex\\_cell](#) const &a, [hex\\_cell](#) const &b)  
*Get the distance between 2 hex cells.*
- [hex\\_cell lava::hex\\_round](#) ([hex\\_frac\\_cell](#) const &cell)  
*Round a fractional cell to hex cell.*
- [hex\\_frac\\_cell lava::hex\\_lerp](#) ([hex\\_frac\\_cell](#) const &a, [hex\\_frac\\_cell](#) const &b, [r32](#) t)  
*Get the linear interpolation between 2 hex cells.*
- [hex\\_cell::list lava::hex\\_line](#) ([hex\\_cell](#) const &a, [hex\\_cell](#) const &b)  
*Get the line between 2 hex cells.*
- [hex\\_offset\\_coord lava::hex\\_q\\_offset\\_from\\_cube](#) ([hex\\_offset](#) offset, [hex\\_cell](#) const &cell)  
*Get the Q offset from hex cube.*
- [hex\\_cell lava::hex\\_q\\_offset\\_to\\_cube](#) ([hex\\_offset](#) offset, [hex\\_offset\\_coord](#) const &coord)  
*Get the Q offset to hex cube.*
- [hex\\_offset\\_coord lava::hex\\_r\\_offset\\_from\\_cube](#) ([hex\\_offset](#) offset, [hex\\_cell](#) const &cell)  
*Get the R offset from hex cube.*
- [hex\\_cell lava::hex\\_r\\_offset\\_to\\_cube](#) ([hex\\_offset](#) offset, [hex\\_offset\\_coord](#) const &coord)  
*Get the R offset to hex cube.*
- [hex\\_doubled\\_coord lava::hex\\_q\\_doubled\\_from\\_cube](#) ([hex\\_cell](#) const &cell)  
*Get the Q doubled from hex cube.*
- [hex\\_cell lava::hex\\_q\\_doubled\\_to\\_cube](#) ([hex\\_doubled\\_coord](#) const &coord)  
*Get the Q doubled to hex cube.*
- [hex\\_doubled\\_coord lava::hex\\_r\\_doubled\\_from\\_cube](#) ([hex\\_cell](#) const &cell)  
*Get the R offset from hex cube.*
- [hex\\_cell lava::hex\\_r\\_doubled\\_to\\_cube](#) ([hex\\_doubled\\_coord](#) const &coord)  
*Get the R doubled to hex cube.*
- [hex\\_point lava::hex\\_to\\_pixel](#) ([hex\\_layout](#) const &layout, [hex\\_cell](#) const &cell)  
*Convert the hex cell to pixel.*
- [hex\\_frac\\_cell lava::hex\\_pixel\\_to\\_cell](#) ([hex\\_layout](#) const &layout, [hex\\_point](#) const &p)  
*Convert the hex point to cell.*
- [hex\\_point lava::hex\\_corner\\_offset](#) ([hex\\_layout](#) const &layout, [i32](#) corner)  
*Get the hex corner offset.*
- [hex\\_point::list lava::hex\\_polygon\\_corners](#) ([hex\\_layout](#) const &layout, [hex\\_cell](#) const &cell)  
*Get the hex polygon corners.*
- [hex\\_point lava::hex\\_get\\_corner](#) ([hex\\_point](#) const &center, [r32](#) size, [ui32](#) corner)  
*Get the hex point by corner.*
- [string lava::to\\_string](#) ([hex\\_cardinal\\_direction](#) direction)  
*Convert hex cardinal direction to string.*
- [hex\\_cell lava::hex\\_get](#) ([hex\\_cardinal\\_direction](#) direction)  
*Get the hex cell from cardinal direction.*
- [hex\\_cardinal\\_direction lava::hex\\_opposite](#) ([hex\\_cardinal\\_direction](#) direction)  
*Get the opposite cardinal direction.*
- [r32 lava::hex\\_calculate\\_inner\\_radius](#) ([r32](#) outer\_radius)  
*Get the hex inner radius from outer radius.*

## Variables

- `hex_cell::list` const `lava::hex_directions`  
*List of hex directions.*
- `hex_cell::list` const `lava::hex_diagonals`  
*List of hex diagonals.*
- `hex_orientation` const `lava::hex_layout_point_y`  
*Hex point Y layout.*
- `hex_orientation` const `lava::hex_layout_flat`  
*Hex flat layout.*
- `hex_cell::list` const `lava::hex_cardinal_directions`  
*List of hex cardinal directions.*
- constexpr `r32` const `lava::hex_inner_radius_factor` = 0.866025404f  
*Hex inner radius factor =  $\sqrt{3} / 2$ .*
- constexpr `r32` const `lava::hex_default_outer_radius` = 1.f  
*Hex default outer radius.*

## 5.140.1 Detailed Description

Hex point, cell and grid.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

### See also

<https://www.redblobgames.com/grids/hexagons/>

## 5.140.2 Function Documentation

### 5.140.2.1 hex\_calculate\_inner\_radius()

```
r32 lava::hex_calculate_inner_radius (
    r32 outer_radius) [inline]
```

Get the hex inner radius from outer radius.

#### Parameters

<code>outer_radius</code>	Hex outer radius
---------------------------	------------------

#### Returns

r32 Hex inner radius

### 5.140.2.2 hex\_cell\_from\_pair()

```
hex_cell lava::hex_cell_from_pair (
    hex_cell::pair const & pair) [inline]
```

Get hex cell from pair.

## Parameters

<i>pair</i>	Hex pair
-------------	----------

## Returns

hex\_cell Hex cell

**5.140.2.3 hex\_corner\_offset()**

```
hex_point lava::hex_corner_offset (  
    hex_layout const & layout,  
    i32 corner) [inline]
```

Get the hex corner offset.

## Parameters

<i>layout</i>	Hex layout
<i>corner</i>	Corner

## Returns

hex\_point Hex point

**5.140.2.4 hex\_diagonal()**

```
hex_cell lava::hex_diagonal (  
    index direction) [inline]
```

Get the diagonal from direction.

## Parameters

<i>direction</i>	Direction index
------------------	-----------------

## Returns

hex\_cell Hex cell

**5.140.2.5 hex\_diagonal\_neighbor()**

```
hex_cell lava::hex_diagonal_neighbor (  
    hex_cell const & cell,  
    index direction) [inline]
```

Get the diagonal neighbor of hex cell by direction.

**Parameters**

<i>cell</i>	Target hex cell
<i>direction</i>	Direction index

**Returns**

hex\_cell Diagonal neighbor hex cell

**5.140.2.6 hex\_direction()**

```
hex_cell lava::hex_direction (
    index direction) [inline]
```

Get the hex cell from direction.

**Parameters**

<i>direction</i>	Direction index
------------------	-----------------

**Returns**

hex\_cell Hex cell

**5.140.2.7 hex\_distance()**

```
i32 lava::hex_distance (
    hex_cell const & a,
    hex_cell const & b) [inline]
```

Get the distance between 2 hex cells.

**Parameters**

<i>a</i>	Source hex cell
<i>b</i>	Target hex cell

**Returns**

i32 Distance

**5.140.2.8 hex\_get()**

```
hex_cell lava::hex_get (
    hex_cardinal_direction direction) [inline]
```

Get the hex cell from cardinal direction.

## Parameters

<i>direction</i>	Hex cardinal direction
------------------	------------------------

## Returns

hex\_cell Hex cell

**5.140.2.9 hex\_get\_corner()**

```
hex_point lava::hex_get_corner (
    hex_point const & center,
    r32 size,
    ui32 corner) [inline]
```

Get the hex point by corner.

## Parameters

<i>center</i>	Center hex point
<i>size</i>	Size of hex cell
<i>corner</i>	Corner

## Returns

hex\_point Hex point

**5.140.2.10 hex\_get\_s()**

```
i32 lava::hex_get_s (
    i32 q,
    i32 r) [inline]
```

Get S axis from Q and R axes.

## Parameters

<i>q</i>	Q axis
<i>r</i>	R axis

## Returns

i32 S axis

**5.140.2.11 hex\_is\_valid()**

```
bool lava::hex_is_valid (
    hex_cell const & cell) [inline]
```

Check if hex cell is valid.

## Parameters

<i>cell</i>	Hex cell to check
-------------	-------------------

## Returns

Hex cell is valid or not

**5.140.2.12 hex\_length()**

```
i32 lava::hex_length (
    hex_cell const & cell) [inline]
```

Get the length of hex cell.

## Parameters

<i>cell</i>	Target hex cell
-------------	-----------------

## Returns

i32 Length of hex cell

**5.140.2.13 hex\_lerp()**

```
hex_frac_cell lava::hex_lerp (
    hex_frac_cell const & a,
    hex_frac_cell const & b,
    r32 t) [inline]
```

Get the linear interpolation between 2 hex cells.

## Parameters

<i>a</i>	Source fractional hex cell
<i>b</i>	Target fractional hex cell
<i>t</i>	Factor

## Returns

hex\_frac\_cell Fractional hex cell

**5.140.2.14 hex\_line()**

```
hex_cell::list lava::hex_line (
    hex_cell const & a,
    hex_cell const & b) [inline]
```

Get the line between 2 hex cells.

## Parameters

<i>a</i>	Source hex cell
<i>b</i>	Target hex cell

## Returns

hex\_cell::list List of hex cells

**5.140.2.15 hex\_neighbor()**

```
hex_cell lava::hex_neighbor (  
    hex_cell const & cell,  
    index direction) [inline]
```

Get the neighbor of hex cell by direction.

## Parameters

<i>cell</i>	Target hex cell
<i>direction</i>	Direction index

## Returns

hex\_cell Neighbor hex cell

**5.140.2.16 hex\_opposite()**

```
hex_cardinal_direction lava::hex_opposite (  
    hex_cardinal_direction direction) [inline]
```

Get the opposite cardinal direction.

## Parameters

<i>direction</i>	Hex cardinal direction
------------------	------------------------

## Returns

hex\_cardinal\_direction Hex cardinal direction

**5.140.2.17 hex\_pixel\_to\_cell()**

```
hex_frac_cell lava::hex_pixel_to_cell (  
    hex_layout const & layout,  
    hex_point const & p) [inline]
```

Convert the hex point to cell.

## Parameters

<i>layout</i>	Hex layout
<i>p</i>	Hex point

## Returns

hex\_frac\_cell Hex fractional cell

**5.140.2.18 hex\_polygon\_corners()**

```
hex_point::list lava::hex_polygon_corners (  
    hex_layout const & layout,  
    hex_cell const & cell) [inline]
```

Get the hex polygon corners.

## Parameters

<i>layout</i>	Hex layout
<i>cell</i>	Hex cell

## Returns

hex\_point::list List of hex points

**5.140.2.19 hex\_q\_doubled\_from\_cube()**

```
hex_doubled_coord lava::hex_q_doubled_from_cube (  
    hex_cell const & cell) [inline]
```

Get the Q doubled from hex cube.

## Parameters

<i>cell</i>	Hex cell
-------------	----------

## Returns

hex\_doubled\_coord Hex doubled coordinates

**5.140.2.20 hex\_q\_doubled\_to\_cube()**

```
hex_cell lava::hex_q_doubled_to_cube (  
    hex_doubled_coord const & coord) [inline]
```

Get the Q doubled to hex cube.



## Parameters

<i>coord</i>	Hex doubled coordinates
--------------	-------------------------

## Returns

hex\_cell Hex cell

**5.140.2.21 hex\_q\_offset\_from\_cube()**

```
hex_offset_coord lava::hex_q_offset_from_cube (
    hex_offset offset,
    hex_cell const & cell) [inline]
```

Get the Q offset from hex cube.

## Parameters

<i>offset</i>	Hex offset
<i>cell</i>	Hex cell

## Returns

hex\_offset\_coord Hex offset coordinates

**5.140.2.22 hex\_q\_offset\_to\_cube()**

```
hex_cell lava::hex_q_offset_to_cube (
    hex_offset offset,
    hex_offset_coord const & coord) [inline]
```

Get the Q offset to hex cube.

## Parameters

<i>offset</i>	Hex offset
<i>coord</i>	Hex offset coordinates

## Returns

hex\_cell Hex cell

**5.140.2.23 hex\_r\_doubled\_from\_cube()**

```
hex_doubled_coord lava::hex_r_doubled_from_cube (
    hex_cell const & cell) [inline]
```

Get the R offset from hex cube.

## Parameters

<i>cell</i>	Hex cell
-------------	----------

## Returns

hex\_doubled\_coord Hex doubled coordinates

**5.140.2.24 hex\_r\_doubled\_to\_cube()**

```
hex_cell lava::hex_r_doubled_to_cube (
    hex_doubled_coord const & coord) [inline]
```

Get the R doubled to hex cube.

## Parameters

<i>coord</i>	Hex doubled coordinates
--------------	-------------------------

## Returns

hex\_cell Hex cell

**5.140.2.25 hex\_r\_offset\_from\_cube()**

```
hex_offset_coord lava::hex_r_offset_from_cube (
    hex_offset offset,
    hex_cell const & cell) [inline]
```

Get the R offset from hex cube.

## Parameters

<i>offset</i>	Hex offset
<i>cell</i>	Hex cell

## Returns

hex\_offset\_coord Hex offset coordinates

**5.140.2.26 hex\_r\_offset\_to\_cube()**

```
hex_cell lava::hex_r_offset_to_cube (
    hex_offset offset,
    hex_offset_coord const & coord) [inline]
```

Get the R offset to hex cube.

## Parameters

<i>offset</i>	Hex offset
<i>coord</i>	Hex offset coordinates

## Returns

hex\_cell Hex cell

**5.140.2.27 hex\_round()**

```
hex_cell lava::hex_round (  
    hex_frac_cell const & cell) [inline]
```

Round a fractional cell to hex cell.

## Parameters

<i>cell</i>	Target fractional cell
-------------	------------------------

## Returns

hex\_cell Rounded hex cell

**5.140.2.28 hex\_to\_pixel()**

```
hex_point lava::hex_to_pixel (  
    hex_layout const & layout,  
    hex_cell const & cell) [inline]
```

Convert the hex cell to pixel.

## Parameters

<i>layout</i>	Hex layout
<i>cell</i>	Hex cell

## Returns

hex\_point Hex point

**5.140.2.29 to\_string()**

```
string lava::to_string (  
    hex_cardinal_direction direction) [inline]
```

Convert hex cardinal direction to string.

**Parameters**

<i>direction</i>	Hex cardinal direction
------------------	------------------------

**Returns**

string String representation

**5.140.3 Variable Documentation****5.140.3.1 hex\_cardinal\_directions**

```
hex_cell::list const lava::hex_cardinal_directions
```

**Initial value:**

```
{
    {1, 0, -1},
    {0, 1, -1},
    {-1, 1, 0},
    {-1, 0, 1},
    {0, -1, 1},
    {1, -1, 0}}
```

List of hex cardinal directions.

**5.140.3.2 hex\_diagonals**

```
hex_cell::list const lava::hex_diagonals
```

**Initial value:**

```
{
    {2, -1, -1},
    {-1, -2, 1},
    {-1, -1, 2},
    {-2, 1, 1},
    {-1, 2, -1},
    {1, 1, -2}}
```

List of hex diagonals.

**5.140.3.3 hex\_directions**

```
hex_cell::list const lava::hex_directions
```

**Initial value:**

```
{
    {1, 0, -1},
    {1, -1, 0},
    {0, -1, 1},
    {-1, 0, 1},
    {-1, 1, 0},
    {0, 1, -1}}
```

List of hex directions.

### 5.140.3.4 hex\_layout\_flat

```
hex_orientation const lava::hex_layout_flat
```

#### Initial value:

```
= {
    3.f / 2.f,
    0.f,
    std::sqrt(3.f) / 2.f,
    std::sqrt(3.f),
    2.f / 3.f,
    0.f,
    -1.f / 3.f,
    std::sqrt(3.f) / 3.f,
    0.f}
```

Hex flat layout.

### 5.140.3.5 hex\_layout\_point\_y

```
hex_orientation const lava::hex_layout_point_y
```

#### Initial value:

```
= {
    std::sqrt(3.f),
    std::sqrt(3.f) / 2.f,
    0.f,
    3.f / 2.f,
    std::sqrt(3.f) / 3.f,
    -1.f / 3.f,
    0.f,
    2.f / 3.f,
    0.5f}
```

Hex point Y layout.

## 5.141 hex.hpp

[Go to the documentation of this file.](#)

```
00001
00009 #pragma once
00010
00011 #include "liblava/core/types.hpp"
00012 #include <cmath>
00013 #include <numbers>
00014 #include <unordered_map>
00015
00016 namespace lava {
00017
00021 struct hex_point {
00023     using list = std::vector<hex_point>;
00024
00026     r32 x{};
00027
00029     r32 y{};
00030 };
00031
00035 struct hex_cell {
00037     using list = std::vector<hex_cell>;
00038
00040     i32 q{};
00041
00043     i32 r{};
00044
00046     i32 s{};
00047
00051     auto operator<=>(hex_cell const&) const = default;
00052
00054     using pair = std::pair<i32, i32>;
00055
00057     using map = std::unordered_map<pair, index, pair_hash>;
```

```

00058
00063     inline pair to_pair() const {
00064         return {q, r};
00065     }
00066
00071     inline void add(hex_cell const& cell) {
00072         *this = {q + cell.q,
00073                 r + cell.r,
00074                 s + cell.s};
00075     }
00076
00081     inline void subtract(hex_cell const& cell) {
00082         *this = {q - cell.q,
00083                 r - cell.r,
00084                 s - cell.s};
00085     }
00086
00091     inline void scale(i32 factor) {
00092         *this = {q * factor,
00093                 r * factor,
00094                 s * factor};
00095     }
00096
00100     inline void rotate_left() {
00101         *this = {-s, -q, -r};
00102     }
00103
00107     inline void rotate_right() {
00108         *this = {-r, -s, -q};
00109     }
00110 };
00111
00118 inline i32 hex_get_s(i32 q, i32 r) {
00119     return -q - r;
00120 }
00121
00127 inline hex_cell hex_cell_from_pair(hex_cell::pair const& pair) {
00128     return {pair.first, pair.second,
00129             hex_get_s(pair.first, pair.second)};
00130 }
00131
00135 struct hex_fractional_cell {
00137     r32 q{};
00138
00140     r32 r{};
00141
00143     r32 s{};
00144 };
00145
00147 using hex_frac_cell = hex_fractional_cell;
00148
00154 inline bool hex_is_valid(hex_cell const& cell) {
00155     return cell.q + cell.r + cell.s == 0;
00156 }
00157
00161 struct hex_offset_coord {
00163     i32 col{};
00164
00166     i32 row{};
00167 };
00168
00170 using hex_doubled_coord = hex_offset_coord;
00171
00175 struct hex_orientation {
00177     r32 f0{};
00178
00180     r32 f1{};
00181
00183     r32 f2{};
00184
00186     r32 f3{};
00187
00189     r32 b0{};
00190
00192     r32 b1{};
00193
00195     r32 b2{};
00196
00198     r32 b3{};
00199
00201     r32 start_angle{};
00202 };
00203
00207 struct hex_layout {
00209     hex_orientation orientation;
00210
00212     hex_point origin;

```

```

00213
00215     hex_point size;
00216 };
00217
00219 hex_cell::list const hex_directions{
00220     {1, 0, -1},
00221     {1, -1, 0},
00222     {0, -1, 1},
00223     {-1, 0, 1},
00224     {-1, 1, 0},
00225     {0, 1, -1}};
00226
00232 inline hex_cell hex_direction(index direction) {
00233     return hex_directions[direction];
00234 }
00235
00242 inline hex_cell hex_neighbor(hex_cell const& cell,
00243                               index direction) {
00244     auto neighbor = cell;
00245     neighbor.add(hex_direction(direction));
00246     return neighbor;
00247 }
00248
00250 hex_cell::list const hex_diagonals{
00251     {2, -1, -1},
00252     {-1, -2, 1},
00253     {-1, -1, 2},
00254     {-2, 1, 1},
00255     {-1, 2, -1},
00256     {1, 1, -2}};
00257
00263 inline hex_cell hex_diagonal(index direction) {
00264     return hex_diagonals[direction];
00265 }
00266
00273 inline hex_cell hex_diagonal_neighbor(hex_cell const& cell,
00274                                         index direction) {
00275     auto neighbor = cell;
00276     neighbor.add(hex_diagonal(direction));
00277     return neighbor;
00278 }
00279
00285 inline i32 hex_length(hex_cell const& cell) {
00286     return to_i32(std::abs(cell.q)
00287                  + std::abs(cell.r)
00288                  + std::abs(cell.s) / 2);
00289 }
00290
00297 inline i32 hex_distance(hex_cell const& a,
00298                           hex_cell const& b) {
00299     auto dist = a;
00300     dist.subtract(b);
00301     return hex_length(dist);
00302 }
00303
00309 inline hex_cell hex_round(hex_frac_cell const& cell) {
00310     auto qi = to_i32(std::round(cell.q));
00311     auto ri = to_i32(std::round(cell.r));
00312     auto si = to_i32(std::round(cell.s));
00313
00314     auto const q_diff = std::abs(qi - cell.q);
00315     auto const r_diff = std::abs(ri - cell.r);
00316     auto const s_diff = std::abs(si - cell.s);
00317
00318     if ((q_diff > r_diff) && (q_diff > s_diff))
00319         qi = -ri - si;
00320     else if (r_diff > s_diff)
00321         ri = -qi - si;
00322     else
00323         si = -qi - ri;
00324
00325     return {qi, ri, si};
00326 }
00327
00335 inline hex_frac_cell hex_lerp(hex_frac_cell const& a,
00336                                hex_frac_cell const& b,
00337                                r32 t) {
00338     return {
00339         a.q * (1.f - t) + b.q * t,
00340         a.r * (1.f - t) + b.r * t,
00341         a.s * (1.f - t) + b.s * t};
00342 }
00343
00350 inline hex_cell::list hex_line(hex_cell const& a,
00351                                 hex_cell const& b) {
00352     auto const a_nudge = hex_frac_cell{a.q + 0.000001f,
00353                                           a.r + 0.000001f,

```

```

00354         a.s - 0.000002f});
00355     auto const b_nudge = hex_frac_cell{b.q + 0.000001f,
00356         b.r + 0.000001f,
00357         b.s - 0.000002f};
00358
00359     ui32 const n = hex_distance(a, b);
00360     auto const step = 1.f / std::max(n, 1u);
00361
00362     hex_cell::list results;
00363     for (auto i = 0u; i <= n; ++i)
00364         results.push_back(hex_round(hex_lerp(a_nudge,
00365             b_nudge,
00366             step * i)));
00367
00368     return results;
00369 }
00370
00371 enum class hex_offset : i32 {
00372     odd = -1,
00373     even = 1
00374 };
00375
00376 inline hex_offset_coord hex_q_offset_from_cube(hex_offset offset,
00377     hex_cell const& cell) {
00378     auto const& col = cell.q;
00379     auto const row = cell.r
00380         + to_i32((cell.q + (i32)offset * (cell.q & 1)) / 2);
00381     return {col, row};
00382 }
00383
00384 inline hex_cell hex_q_offset_to_cube(hex_offset offset,
00385     hex_offset_coord const& coord) {
00386     auto const& q = coord.col;
00387     auto const r = coord.row
00388         - to_i32((coord.col + (i32)offset * (coord.col & 1)) / 2);
00389     auto const s = -q - r;
00390     return {q, r, s};
00391 }
00392
00393 inline hex_offset_coord hex_r_offset_from_cube(hex_offset offset,
00394     hex_cell const& cell) {
00395     auto const col = cell.q
00396         + to_i32((cell.r + (i32)offset * (cell.r & 1)) / 2);
00397     auto const& row = cell.r;
00398     return {col, row};
00399 }
00400
00401 inline hex_cell hex_r_offset_to_cube(hex_offset offset,
00402     hex_offset_coord const& coord) {
00403     auto const q = coord.col
00404         - to_i32((coord.row + (i32)offset * (coord.row & 1)) / 2);
00405     auto const& r = coord.row;
00406     auto const s = -q - r;
00407     return {q, r, s};
00408 }
00409
00410 inline hex_doubled_coord hex_q_doubled_from_cube(hex_cell const& cell) {
00411     auto const& col = cell.q;
00412     auto const row = 2 * cell.r + cell.q;
00413     return {col, row};
00414 }
00415
00416 inline hex_cell hex_q_doubled_to_cube(hex_doubled_coord const& coord) {
00417     auto const& q = coord.col;
00418     auto const r = to_i32((coord.row - coord.col) / 2);
00419     auto const s = -q - r;
00420     return {q, r, s};
00421 }
00422
00423 inline hex_doubled_coord hex_r_doubled_from_cube(hex_cell const& cell) {
00424     auto const col = 2 * cell.q + cell.r;
00425     auto const& row = cell.r;
00426     return {col, row};
00427 }
00428
00429 inline hex_cell hex_r_doubled_to_cube(hex_doubled_coord const& coord) {
00430     auto const q = to_i32((coord.col - coord.row) / 2);
00431     auto const& r = coord.row;
00432     auto const s = -q - r;
00433     return {q, r, s};
00434 }
00435
00436 hex_orientation const hex_layout_point_y = {
00437     std::sqrt(3.f),
00438     std::sqrt(3.f) / 2.f,
00439     0.f,
00440     3.f / 2.f,

```



```

00489     std::sqrt(3.f) / 3.f,
00490     -1.f / 3.f,
00491     0.f,
00492     2.f / 3.f,
00493     0.5f};
00494
00495 hex_orientation const hex_layout_flat = {
00496     3.f / 2.f,
00497     0.f,
00498     std::sqrt(3.f) / 2.f,
00499     std::sqrt(3.f),
00500     2.f / 3.f,
00501     0.f,
00502     -1.f / 3.f,
00503     std::sqrt(3.f) / 3.f,
00504     0.f};
00505
00506 inline hex_point hex_to_pixel(hex_layout const& layout,
00507     hex_cell const& cell) {
00508     auto const& m = layout.orientation;
00509     auto const& size = layout.size;
00510     auto const& origin = layout.origin;
00511     auto const x = (m.f0 * cell.q + m.f1 * cell.r) * size.x;
00512     auto const y = (m.f2 * cell.q + m.f3 * cell.r) * size.y;
00513     return {x + origin.x, y + origin.y};
00514 }
00515
00516 inline hex_frac_cell hex_pixel_to_cell(hex_layout const& layout,
00517     hex_point const& p) {
00518     auto const& m = layout.orientation;
00519     auto const& size = layout.size;
00520     auto const& origin = layout.origin;
00521     auto const pt = hex_point{(p.x - origin.x) / size.x,
00522         (p.y - origin.y) / size.y};
00523     auto const q = m.b0 * pt.x + m.b1 * pt.y;
00524     auto const r = m.b2 * pt.x + m.b3 * pt.y;
00525     return {q, r, -q - r};
00526 }
00527
00528 inline hex_point hex_corner_offset(hex_layout const& layout,
00529     i32 corner) {
00530     auto const& m = layout.orientation;
00531     auto const& size = layout.size;
00532     auto const angle = 2.
00533         * std::numbers::pi_v<r32> * (m.start_angle - corner)
00534         / 6.f;
00535     return {size.x * (r32)std::cos(angle),
00536         size.y * (r32)std::sin(angle)};
00537 }
00538
00539 inline hex_point::list hex_polygon_corners(hex_layout const& layout,
00540     hex_cell const& cell) {
00541     hex_point::list corners = {};
00542     auto const center = hex_to_pixel(layout, cell);
00543     for (int i = 0u; i < 6; ++i) {
00544         auto const offset = hex_corner_offset(layout, i);
00545         corners.push_back({center.x + offset.x,
00546             center.y + offset.y});
00547     }
00548     return corners;
00549 }
00550
00551 inline hex_point hex_get_corner(hex_point const& center,
00552     r32 size,
00553     ui32 corner) {
00554     auto const angle_deg = 60 * corner - 30;
00555     auto const angle_rad = std::numbers::pi_v<r32> / 180 * angle_deg;
00556     return {
00557         center.x + size * std::cos(angle_rad),
00558         center.y + size * std::sin(angle_rad)};
00559 }
00560
00561 enum class hex_cardinal_direction : index {
00562     NE = 0,
00563     E,
00564     SE,
00565     SW,
00566     W,
00567     NW
00568 };
00569
00570 inline string to_string(hex_cardinal_direction direction) {
00571     switch (direction) {
00572     case hex_cardinal_direction::NE: {
00573         return "Northeast";
00574     }
00575     }
00576 }

```

```

00616     case hex_cardinal_direction::E: {
00617         return "East";
00618     }
00619     case hex_cardinal_direction::SE: {
00620         return "Southeast";
00621     }
00622     case hex_cardinal_direction::SW: {
00623         return "Southwest";
00624     }
00625     case hex_cardinal_direction::W: {
00626         return "West";
00627     }
00628     case hex_cardinal_direction::NW: {
00629         return "Northwest";
00630     }
00631     }
00632 }
00633
00635 hex_cell::list const hex_cardinal_directions{
00636     {1, 0, -1},
00637     {0, 1, -1},
00638     {-1, 1, 0},
00639     {-1, 0, 1},
00640     {0, -1, 1},
00641     {1, -1, 0}};
00642
00648 inline hex_cell hex_get(hex_cardinal_direction direction) {
00649     return hex_cardinal_directions[(index)direction];
00650 }
00651
00657 inline hex_cardinal_direction hex_opposite(hex_cardinal_direction direction) {
00658     if ((index)direction < 3)
00659         return hex_cardinal_direction((i32)direction + 3);
00660     else
00661         return hex_cardinal_direction((i32)direction - 3);
00662 }
00663
00665 constexpr r32 const hex_inner_radius_factor = 0.866025404f;
00666
00668 constexpr r32 const hex_default_outer_radius = 1.f;
00669
00675 inline r32 hex_calculate_inner_radius(r32 outer_radius) {
00676     return outer_radius * hex_inner_radius_factor;
00677 }
00678
00682 struct hex_grid {
00683     r32 inner_radius = 0.f;
00684
00685     r32 outer_radius = hex_default_outer_radius;
00686
00687     hex_layout layout;
00688
00696     hex_grid(r32 radius = hex_default_outer_radius)
00697     : outer_radius(radius) {
00698         update();
00699     }
00700
00705     void update(hex_orientation orientation = hex_layout_point_y) {
00706         inner_radius = hex_calculate_inner_radius(outer_radius);
00707         layout = {
00708             orientation,
00709             {},
00710             {outer_radius, outer_radius}};
00711     }
00712
00719     hex_cell find(r32 x, r32 y) const {
00720         return hex_round(hex_pixel_to_cell(layout,
00721             {x, y}));
00722     }
00723
00729     hex_point to_pixel(hex_cell const& cell) const {
00730         return hex_to_pixel(layout, cell);
00731     }
00732 };
00733
00734 } // namespace lava

```

## 5.142 liblava/util/layer.hpp File Reference

Layering.

```
#include "liblava/core/id.hpp"
#include "liblava/core/misc.hpp"
```

## Classes

- struct [lava::layer](#)  
*Layer.*
- struct [lava::layer\\_list](#)  
*Layer list.*

## 5.142.1 Detailed Description

Layering.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.143 layer.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/id.hpp"
00011 #include "liblava/core/misc.hpp"
00012
00013 namespace lava {
00014
00018 struct layer : entity {
00020     using s_ptr = std::shared_ptr<layer>;
00021
00023     using map = std::map<id, s_ptr>;
00024
00026     using list = std::vector<s_ptr>;
00027
00029     using func = std::function<void()>;
00030
00036     static s_ptr make(string_ref name) {
00037         return std::make_shared<layer>(name);
00038     }
00039
00044     layer(string_ref name)
00045     : name(name) {}
00046
00048     func on_func;
00049
00051     bool active = true;
00052
00054     string name;
00055 };
00056
00060 struct layer_list {
00062     using ptr = layer_list*;
00063
00071     id add(string_ref name,
00072           layer::func func,
00073           bool active = true) {
00074         auto layer = layer::make(name);
```

```

00075
00076     layer->on_func = func;
00077     layer->active = active;
00078
00079     m_layers.push_back(layer);
00080
00081     return layer->get_id();
00082 }
00083
00084 void add(layer::s_ptr layer) {
00085     m_layers.push_back(layer);
00086 }
00087
00088 id add_inactive(string_ref name,
00089                 layer::func func) {
00090     return add(name, func, false);
00091 }
00092
00093 layer::s_ptr get(id::ref layer_id) {
00094     for (auto const& layer : m_layers)
00095         if (layer->get_id() == layer_id)
00096             return layer;
00097
00098     return nullptr;
00099 }
00100
00101 bool remove(id::ref layer_id) {
00102     for (auto const& layer : m_layers) {
00103         if (layer->get_id() == layer_id) {
00104             lava::remove(m_layers, layer);
00105             return true;
00106         }
00107     }
00108
00109     return false;
00110 }
00111
00112 layer::list const& get_all() const {
00113     return m_layers;
00114 }
00115
00116 void clear() {
00117     m_layers.clear();
00118 }
00119
00120 private:
00121     layer::list m_layers;
00122 };
00123
00124 } // namespace lava

```

## 5.144 liblava/util/log.hpp File Reference

Logging.

```

#include "liblava/core/version.hpp"
#include "spdlog/sinks/basic_file_sink.h"
#include "spdlog/sinks/stdout_color_sinks.h"
#include "spdlog/spdlog.h"
#include <memory>

```

### Classes

- struct [lava::log::config](#)  
*Log configuration.*
- struct [lava::global\\_logger](#)  
*Global logger.*

## Typedefs

- using **lava::s\_logger** = std::shared\_ptr<spdlog::logger>  
*Logger.*

## Functions

- [string lava::to\\_string \(string\\_ref id, string\\_ref name\)](#)  
*Convert id and name to string.*
- [string lava::to\\_string \(sem\\_version const &version\)](#)  
*Convert semantic version to string.*
- [string lava::semantic\\_version\\_string \(\)](#)  
*Convert global semantic version to string.*
- [string lava::sem\\_version\\_string \(\)](#)
- [name lava::to\\_string \(version\\_stage stage\)](#)  
*Convert version stage to string.*
- [string lava::to\\_string \(version const &version\)](#)  
*Convert version to string.*
- [string lava::version\\_string \(\)](#)  
*Convert global version to string.*
- [s\\_logger lava::log::setup \(config config={}\)](#)  
*Set up logging.*
- void [lava::log::teardown \(config config={}\)](#)  
*Tear down logging.*
- [s\\_logger lava::logger \(\)](#)  
*Get global logger.*

### 5.144.1 Detailed Description

Logging.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.144.2 Function Documentation

#### 5.144.2.1 logger()

```
s_logger lava::logger () [inline]
```

Get global logger.

#### Returns

s\_logger Logger

#### 5.144.2.2 sem\_version\_string()

```
string lava::sem_version_string () [inline]
```

See also

semantic\_version\_string

#### 5.144.2.3 semantic\_version\_string()

```
string lava::semantic_version_string () [inline]
```

Convert global semantic version to string.

Returns

string String representation

#### 5.144.2.4 setup()

```
s_logger lava::log::setup (  
    config config = {}) [inline]
```

Set up logging.

Parameters

<i>config</i>	Log configuration
---------------	-------------------

Returns

s\_logger Logger

#### 5.144.2.5 teardown()

```
void lava::log::teardown (  
    config config = {}) [inline]
```

Tear down logging.

Parameters

<i>config</i>	Log configuration
---------------	-------------------

#### 5.144.2.6 to\_string() [1/4]

```
string lava::to_string (  
    sem_version const & version) [inline]
```

Convert semantic version to string.

## Parameters

<i>version</i>	Semantic version to convert
----------------	-----------------------------

## Returns

string String representation

**5.144.2.7 to\_string() [2/4]**

```
string lava::to_string (  
    string_ref id,  
    string_ref name) [inline]
```

Convert id and name to string.

## Parameters

<i>id</i>	Id to convert
<i>name</i>	Name to convert

## Returns

string String representation

**5.144.2.8 to\_string() [3/4]**

```
string lava::to_string (  
    version const & version) [inline]
```

Convert version to string.

## Parameters

<i>version</i>	Version to convert
----------------	--------------------

## Returns

string String representation

**5.144.2.9 to\_string() [4/4]**

```
name lava::to_string (  
    version_stage stage) [inline]
```

Convert version stage to string.

## Parameters

<i>stage</i>	Version stage to convert
--------------	--------------------------

## Returns

name Name representation

## 5.144.2.10 version\_string()

```
string lava::version_string () [inline]
```

Convert global version to string.

## Returns

string String representation

## 5.145 log.hpp

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/version.hpp"
00011 #include "spdlog/sinks/basic_file_sink.h"
00012 #include "spdlog/sinks/stdout_color_sinks.h"
00013 #include "spdlog/spdlog.h"
00014 #include <memory>
00015
00016 namespace lava {
00017
00019 using s_logger = std::shared_ptr<spdlog::logger>;
00020
00027 inline string to_string(string_ref id, string_ref name) {
00028     return fmt::format("{} | {}", id, name);
00029 }
00030
00036 inline string to_string(sem_version const& version) {
00037     return fmt::format("{}.{}.{}",
00038         version.major,
00039         version.minor,
00040         version.patch);
00041 }
00042
00047 inline string semantic_version_string() {
00048     return to_string(sem_version{});
00049 }
00050
00054 inline string sem_version_string() {
00055     return semantic_version_string();
00056 }
00057
00063 inline name to_string(version_stage stage) {
00064     switch (stage) {
00065     case version_stage::preview:
00066         return "preview";
00067     case version_stage::alpha:
00068         return "alpha";
00069     case version_stage::beta:
00070         return "beta";
00071     case version_stage::rc:
00072         return "rc";
00073     default:
00074         return "";
00075     }
00076 }
00077
```



```

00083 inline string to_string(version const& version) {
00084     string stage_str = to_string(version.stage);
00085     if ((version.rev > 1) && (version.stage != version_stage::release))
00086         stage_str += fmt::format(" {}", version.rev);
00087
00088     if (version.release == 0) {
00089         if (stage_str.empty())
00090             return fmt::format("{}", version.year);
00091         else
00092             return fmt::format("{} {}",
00093                                 version.year, stage_str);
00094     } else
00095         return fmt::format("{}.{ } {}",
00096                             version.year, version.release, stage_str);
00097 }
00098
00103 inline string version_string() {
00104     return to_string(version{});
00105 }
00106
00107 namespace log {
00108
00112     struct config {
00113         name logger = _lava_;
00114
00115         name file = "lava.log";
00116
00117         i32 level = undef;
00118
00119         bool debug = false;
00120     };
00121
00122     inline s_logger setup(config config = {}) {
00123         if (config.debug) {
00124             auto log = spdlog::stdout_color_mt(config.logger);
00125             log->set_level((config.level < 0)
00126                          ? spdlog::level::debug
00127                          : (spdlog::level::level_enum)config.level);
00128             return log;
00129         } else {
00130             auto log = spdlog::basic_logger_mt(config.logger, config.file);
00131             log->set_level((config.level < 0)
00132                          ? spdlog::level::warn
00133                          : (spdlog::level::level_enum)config.level);
00134             return log;
00135         }
00136     }
00137
00138     inline void teardown(config config = {}) {
00139         spdlog::drop(config.logger);
00140     }
00141
00142 } // namespace log
00143
00144 struct global_logger {
00145     static global_logger& singleton() {
00146         static global_logger global_logger;
00147         return global_logger;
00148     }
00149
00150     s_logger get() {
00151         return m_logger;
00152     }
00153
00154     void set(lava::s_logger l) {
00155         m_logger = l;
00156     }
00157
00158     void reset() {
00159         m_logger = nullptr;
00160     }
00161
00162 private:
00163     s_logger m_logger;
00164 };
00165
00166 inline s_logger logger() {
00167     return global_logger::singleton().get();
00168 }
00169
00170 } // namespace lava

```

## 5.146 liblava/util/math.hpp File Reference

Math helpers.

```
#include "liblava/core/types.hpp"
#include "picosha2.h"
#include "glm/glm.hpp"
#include "glm/gtc/matrix_transform.hpp"
#include "glm/gtc/type_ptr.hpp"
```

### Classes

- struct [lava::rect](#)  
*Rectangle.*

### Typedefs

- using **lava::v2** = glm::vec2  
*Vector 2D.*
- using **lava::v3** = glm::vec3  
*Vector 3D.*
- using **lava::v4** = glm::vec4  
*Vector 4D.*
- using **lava::uv2** = glm::uvec2  
*UV pair.*
- using **lava::mat3** = glm::mat3  
*Matrix 3x3.*
- using **lava::mat4** = glm::mat4  
*Matrix 4x4.*
- using **lava::iv2** = glm::ivec2  
*Integer vector 2D.*
- using **lava::iv3** = glm::ivec3  
*Integer vector 3D.*

### Functions

- auto [lava::ceil\\_div](#) (auto x, auto y)  
*Ceiling of division.*
- [mat4 lava::perspective\\_matrix](#) (uv2 size, [r32](#) fov=90.f, [r32](#) far\_plane=5.f)  
*Calculate perspective matrix.*
- [string lava::hash256](#) ([string\\_ref](#) value)  
*Get SHA-256 hash of string.*

### Variables

- [v3](#) const [lava::default\\_color](#)  
*Default color (Lava color: CF1020 : 207, 16, 32)*

## 5.146.1 Detailed Description

Math helpers.

### Authors

Lava Block OÜ and contributors

### Copyright

Copyright (c) 2018-present, MIT License

## 5.146.2 Function Documentation

### 5.146.2.1 `ceil_div()`

```
auto lava::ceil_div (  
    auto x,  
    auto y) [inline]
```

Ceiling of division.

#### Parameters

<i>x</i>	X value
<i>y</i>	Y value

#### Returns

auto Result

### 5.146.2.2 `hash256()`

```
string lava::hash256 (  
    string_ref value) [inline]
```

Get SHA-256 hash of string.

#### Parameters

<i>value</i>	Value to hash
--------------	---------------

#### Returns

string Hash result

### 5.146.2.3 `perspective_matrix()`

```
mat4 lava::perspective_matrix (  
    uv2 size,  
    r32 fov = 90.f,  
    r32 far_plane = 5.f) [inline]
```

Calculate perspective matrix.

**Parameters**

<i>size</i>	Size for aspect ratio
<i>fov</i>	Field of view
<i>far_plane</i>	Far plane

**Returns**

mat4 Calculated matrix

**5.146.3 Variable Documentation****5.146.3.1 default\_color**

```
v3 const lava::default_color
```

**Initial value:**

```
= v3{0.8118f,
      0.0627f,
      0.1255f}
```

Default color (Lava color: CF1020 : 207, 16, 32)

**5.147 math.hpp**

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include "picosha2.h"
00012
00013 #define GLM_FORCE_RADIANS
00014 #define GLM_FORCE_DEPTH_ZERO_TO_ONE
00015 #include "glm/glm.hpp"
00016 #include "glm/gtc/matrix_transform.hpp"
00017 #include "glm/gtc/type_ptr.hpp"
00018
00019 namespace lava {
00020
00022 using v2 = glm::vec2;
00023
00025 using v3 = glm::vec3;
00026
00028 using v4 = glm::vec4;
00029
00031 using uv2 = glm::uvec2;
00032
00034 using mat3 = glm::mat3;
00035
00037 using mat4 = glm::mat4;
00038
00040 using iv2 = glm::ivec2;
00041
00043 using iv3 = glm::ivec3;
00044
00048 struct rect {
00050     using ref = rect const&;
00051
00055     rect() = default;
00056
00064     rect(i32 left, i32 top,
00065         ui32 width, ui32 height)
00066     : m_left_top({left, top}) {
00067         set_size({width, height});
00068     }
```

```

00069
00076     rect(iv2 const& left_top,
00077          ui32 width, ui32 height)
00078     : m_left_top(left_top) {
00079         set_size({width, height});
00080     }
00081
00087     rect(iv2 const& left_top,
00088          uv2 const& size)
00089     : m_left_top(left_top) {
00090         set_size(size);
00091     }
00092
00097     iv2 const& get_origin() const {
00098         return m_left_top;
00099     }
00100
00105     iv2 const& get_end_point() const {
00106         return m_right_bottom;
00107     }
00108
00113     uv2 get_size() const {
00114         LAVA_ASSERT(m_left_top.x <= m_right_bottom.x);
00115         LAVA_ASSERT(m_left_top.y <= m_right_bottom.y);
00116         return {m_right_bottom.x - m_left_top.x,
00117                 m_right_bottom.y - m_left_top.y};
00118     }
00119
00124     void set_size(uv2 const& size) {
00125         m_right_bottom.x = m_left_top.x + size.x;
00126         m_right_bottom.y = m_left_top.y + size.y;
00127     }
00128
00133     void move(iv2 const& offset) {
00134         m_left_top += offset;
00135         m_right_bottom += offset;
00136     }
00137
00143     bool contains(iv2 point) const {
00144         return (m_left_top.x < point.x)
00145             && (m_left_top.y < point.y)
00146             && (m_right_bottom.x > point.x)
00147             && (m_right_bottom.y > point.y);
00148     }
00149
00150 private:
00152     iv2 m_left_top = iv2();
00153
00155     iv2 m_right_bottom = iv2();
00156 };
00157
00164 inline auto ceil_div(auto x, auto y) {
00165     return (x + y - 1) / y;
00166 }
00167
00169 v3 const default_color = v3{0.8118f,
00170                             0.0627f,
00171                             0.1255f};
00172
00180 inline mat4 perspective_matrix(uv2 size,
00181                                r32 fov = 90.f,
00182                                r32 far_plane = 5.f) {
00183     // Vulkan NDC is right-handed with Y pointing down
00184     // we flip Y which makes it left-handed
00185     return glm::scale(glm::identity<glm::mat4>(),
00186                       {1.f, -1.f, 1.f})
00187         * glm::perspectiveLH_ZO(
00188             glm::radians(fov),
00189             r32(size.x) / size.y,
00190             0.1f,
00191             far_plane);
00192 }
00193
00199 inline string hash256(string_ref value) {
00200     std::vector<uc8> hash(picsha2::k_digest_size);
00201     picsha2::hash256(value.begin(), value.end(),
00202                      hash.begin(), hash.end());
00203
00204     return picsha2::bytes_to_hex_string(hash.begin(), hash.end());
00205 }
00206
00207 } // namespace lava

```

## 5.148 liblava/util/random.hpp File Reference

Random generator.

```
#include "liblava/core/types.hpp"
#include <random>
```

### Classes

- struct [lava::random\\_generator](#)  
*Random generator.*
- struct [lava::pseudorandom\\_generator](#)  
*Pseudorandom generator.*

### Functions

- auto [lava::random](#) (auto low, auto high)  
*Get next random number.*
- auto [lava::random](#) (auto high)  
*Get next random number (lowest is 0)*

### 5.148.1 Detailed Description

Random generator.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.148.2 Function Documentation

#### 5.148.2.1 random() [1/2]

```
auto lava::random (
    auto high) [inline]
```

Get next random number (lowest is 0)

#### Parameters

<i>high</i>	Highest number
-------------	----------------

#### Returns

auto Random number

#### 5.148.2.2 random() [2/2]

```
auto lava::random (
    auto low,
    auto high) [inline]
```

Get next random number.

## Parameters

<i>low</i>	Lowest number
<i>high</i>	Highest number

## Returns

auto Random number

## 5.149 random.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/core/types.hpp"
00011 #include <random>
00012
00013 namespace lava {
00014
00018 struct random_generator {
00022     random_generator() {
00023         std::random_device rd;
00024         m_engine = std::mt19937(rd());
00025     }
00026
00033     i32 get(i32 low, i32 high) {
00034         std::uniform_int_distribution<i32> dist(low, high);
00035         return dist(m_engine);
00036     }
00037
00045     template <typename T = real>
00046     T get(T low, T high) {
00047         std::uniform_real_distribution<T> dist(low, high);
00048         return dist(m_engine);
00049     }
00050
00051 private:
00053     std::mt19937 m_engine;
00054 };
00055
00062 inline auto random(auto low, auto high) {
00063     return random_generator().get(low, high);
00064 }
00065
00071 inline auto random(auto high) {
00072     return random_generator().get({}, high);
00073 }
00074
00078 struct pseudorandom_generator {
00083     explicit pseudorandom_generator(ui32 seed)
00084         : m_seed(seed) {}
00085
00090     void set_seed(ui32 value) {
00091         m_seed = value;
00092     }
00093
00098     ui32 get() {
00099         return generate_fast() ^ (generate_fast() >> 7);
00100     }
00101
00102 private:
00104     ui32 m_seed = 0;
00105
00110     ui32 generate_fast() {
00111         return m_seed = (m_seed * 196314165 + 907633515);
00112     }
00113 };
00114
00115 } // namespace lava

```

## 5.150 liblava/util/telegram.hpp File Reference

Message dispatcher.

```
#include "liblava/util/thread.hpp"
#include <any>
#include <cmath>
#include <set>
```

### Classes

- struct [lava::telegram](#)  
*Telegram.*
- struct [lava::telegraph](#)  
*Telegraph station.*
- struct [lava::message\\_dispatcher](#)  
*Message dispatcher.*

### Typedefs

- using [lava::any](#) = std::any  
*Any type.*

### Variables

- constexpr [ms](#) const [lava::telegram\\_min\\_delay](#) {250}  
*Minimal telegram delay in milliseconds.*

### 5.150.1 Detailed Description

Message dispatcher.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License



## 5.151 telegram.hpp

[Go to the documentation of this file.](#)

```

00001
00008 #pragma once
00009
00010 #include "liblava/util/thread.hpp"
00011 #include <any>
00012 #include <cmath>
00013 #include <set>
00014
00015 namespace lava {
00016
00018 constexpr ms const telegram_min_delay{250};
00019
00021 using any = std::any;
00022
00026 struct telegram {
00028     using ref = telegram const&;
00029
00031     using set = std::multiset<telegram>;
00032
00041     explicit telegram(id::ref sender,
00042                     id::ref receiver,
00043                     index msg,
00044                     ms dispatch_time = {},
00045                     any info = {})
00046     : sender(sender), receiver(receiver),
00047       msg_id(msg), dispatch_time(dispatch_time),
00048       info(std::move(info)) {}
00049
00055     bool operator==(ref rhs) const {
00056         return ((dispatch_time - rhs.dispatch_time) < telegram_min_delay)
00057             && (sender == rhs.sender)
00058             && (receiver == rhs.receiver)
00059             && (msg_id == rhs.msg_id);
00060     }
00061
00067     bool operator<(ref rhs) const {
00068         if (*this == rhs)
00069             return false;
00070
00071         return (dispatch_time < rhs.dispatch_time);
00072     }
00073
00075     id sender;
00076
00078     id receiver;
00079
00081     index msg_id = no_index;
00082
00084     ms dispatch_time;
00085
00087     any info;
00088 };
00089
00093 struct telegraph : interface {
00102     virtual void send_message(id::ref receiver,
00103                             id::ref sender,
00104                             index message,
00105                             ms delay = {},
00106                             any const& info = {}) = 0;
00107 };
00108
00112 struct message_dispatcher : telegraph {
00116     ~message_dispatcher() {
00117         teardown();
00118     }
00119
00124     void setup(ui32 thread_count) {
00125         m_pool.setup(thread_count);
00126     }
00127
00131     void teardown() {
00132         m_pool.teardown();
00133     }
00134
00139     void update(ms current) {
00140         m_current_time = current;
00141         dispatch_delayed_messages(m_current_time);
00142     }
00143
00145     void send_message(id::ref receiver,
00146                     id::ref sender,
00147                     index message,

```

```

00148             ms delay = {},
00149             any const& info = {}) override {
00150         telegram msg(sender,
00151             receiver,
00152             message,
00153             m_current_time,
00154             info);
00155
00156         if (delay == ms{0}) {
00157             discharge(msg); // now
00158             return;
00159         }
00160
00161         msg.dispatch_time += delay;
00162         m_messages.insert(msg);
00163     }
00164
00165     using message_func = std::function<void(telegram::ref, id::ref)>;
00166
00167     bool add_dispatch(id::ref target, message_func func) {
00168         std::lock_guard guard(m_lock);
00169
00170         if (m_dispatches.count(target))
00171             return false;
00172
00173         m_dispatches.emplace(target, func);
00174         return true;
00175     }
00176
00177     bool remove_dispatch(id::ref target) {
00178         std::lock_guard guard(m_lock);
00179
00180         if (!m_dispatches.count(target))
00181             return false;
00182
00183         m_dispatches.erase(target);
00184         return true;
00185     }
00186
00187     bool has_dispatch(id::ref target) const {
00188         return m_dispatches.count(target);
00189     }
00190
00191 private:
00192     void discharge(telegram::ref message) {
00193         m_pool.enqueue([&, message](id::ref thread_id) {
00194             std::lock_guard guard(m_lock);
00195
00196             auto dispatch = get_dispatch(message.receiver);
00197             LAVA_ASSERT(dispatch);
00198             if (dispatch)
00199                 dispatch(message, thread_id);
00200         });
00201     }
00202
00203     void dispatch_delayed_messages(ms time) {
00204         while (!m_messages.empty()
00205             && (m_messages.begin()->dispatch_time < time)) {
00206             discharge(*m_messages.begin());
00207             m_messages.erase(m_messages.begin());
00208         }
00209     }
00210
00211     message_func get_dispatch(id::ref target) {
00212         if (!m_dispatches.count(target))
00213             return nullptr;
00214
00215         return m_dispatches.at(target);
00216     }
00217
00218     using dispatch_map = std::map<id, message_func>;
00219
00220     dispatch_map m_dispatches;
00221
00222     std::mutex m_lock;
00223
00224     ms m_current_time;
00225
00226     thread_pool m_pool;
00227
00228     telegram::set m_messages;
00229 };
00230
00231 } // namespace lava

```

## 5.152 liblava/util/thread.hpp File Reference

Thread pool.

```
#include "liblava/core/id.hpp"
#include "liblava/core/time.hpp"
#include <condition_variable>
#include <deque>
#include <mutex>
#include <thread>
```

### Classes

- struct [lava::thread\\_pool](#)  
*Thread pool.*

### Functions

- void [lava::sleep](#) ([seconds](#) time)  
*Sleep for seconds.*
- void [lava::sleep](#) ([ms](#) time)  
*Sleep for milliseconds.*
- void [lava::sleep](#) ([us](#) time)  
*Sleep for microseconds.*

### 5.152.1 Detailed Description

Thread pool.

#### Authors

Lava Block OÜ and contributors

#### Copyright

Copyright (c) 2018-present, MIT License

### 5.152.2 Function Documentation

#### 5.152.2.1 [sleep\(\)](#) [1/3]

```
void lava::sleep (  
    ms time) [inline]
```

Sleep for milliseconds.

## Parameters

<i>time</i>	Milliseconds to sleep
-------------	-----------------------

**5.152.2.2 sleep() [2/3]**

```
void lava::sleep (
    seconds time) [inline]
```

Sleep for seconds.

## Parameters

<i>time</i>	Seconds to sleep
-------------	------------------

**5.152.2.3 sleep() [3/3]**

```
void lava::sleep (
    us time) [inline]
```

Sleep for microseconds.

## Parameters

<i>time</i>	Microseconds to sleep
-------------	-----------------------

**5.153 thread.hpp**

[Go to the documentation of this file.](#)

```
00001
00008 #pragma once
00009
00010 #include "liblava/core/id.hpp"
00011 #include "liblava/core/time.hpp"
00012 #include <condition_variable>
00013 #include <deque>
00014 #include <mutex>
00015 #include <thread>
00016
00017 namespace lava {
00018
00023 inline void sleep(seconds time) {
00024     std::this_thread::sleep_for(time);
00025 }
00026
00031 inline void sleep(ms time) {
00032     std::this_thread::sleep_for(time);
00033 }
00034
00039 inline void sleep(us time) {
00040     std::this_thread::sleep_for(time);
00041 }
00042
00046 struct thread_pool {
00048     using task = std::function<void(id::ref)>;
00049
00054     void setup(ui32 count = 2) {
00055         for (auto i = 0u; i < count; ++i)
```

```

00056         m_workers.emplace_back(worker(*this));
00057     }
00058
00062     void teardown() {
00063         m_stop = true;
00064         m_condition.notify_all();
00065
00066         for (auto& worker : m_workers)
00067             worker.join();
00068
00069         m_workers.clear();
00070     }
00071
00076     void enqueue(auto f) {
00077         {
00078             std::unique_lock<std::mutex> lock(m_queue_mutex);
00079             m_tasks.push_back(task(f));
00080         }
00081         m_condition.notify_one();
00082     }
00083
00084 private:
00088     struct worker {
00093         explicit worker(thread_pool& pool)
00094             : m_pool(pool) {}
00095
00099         void operator() () {
00100             auto thread_id = ids::instance().next();
00101
00102             task task;
00103             while (true) {
00104                 {
00105                     std::unique_lock<std::mutex> lock(m_pool.m_queue_mutex);
00106
00107                     while (!m_pool.m_stop && m_pool.m_tasks.empty())
00108                         m_pool.m_condition.wait(lock);
00109
00110                     if (m_pool.m_stop)
00111                         break;
00112
00113                     task = m_pool.m_tasks.front();
00114                     m_pool.m_tasks.pop_front();
00115                 }
00116
00117                 task(thread_id);
00118             }
00119         }
00120
00121     private:
00123         thread_pool& m_pool;
00124     };
00125
00127     std::vector<std::thread> m_workers;
00128
00130     std::deque<task> m_tasks;
00131
00133     std::mutex m_queue_mutex;
00134
00136     std::condition_variable m_condition;
00137
00139     bool m_stop = false;
00140 };
00141
00142 } // namespace lava

```



# Index

- activated
  - lava::block, [37](#)
  - lava::camera, [53](#)
  - lava::imgui, [166](#)
  - lava::pipeline, [219](#)
  - lava::subpass, [295](#)
- active
  - lava::key\_event, [183](#)
- add
  - lava::descriptor, [77](#)
  - lava::gamepad\_manager, [137](#)
  - lava::hex\_cell, [140](#)
  - lava::id\_listeners< T >, [149](#)
  - lava::id\_registry< T, Meta >, [151](#)
  - lava::input, [171](#)
  - lava::input\_events< T >, [173](#)
  - lava::json\_file, [180](#)
  - lava::layer\_list, [188](#)
  - lava::pipeline\_layout, [223](#)
  - lava::props, [237](#)
  - lava::queue\_family\_info, [244](#)
  - lava::render\_pass, [252](#), [253](#)
  - lava::render\_pipeline, [262](#)
  - lava::staging, [291](#)
  - lava::subpass, [295](#)
  - lava::tooltip\_list, [324](#)
- add\_binding
  - lava::descriptor, [77](#)
- add\_callback
  - lava::render\_target, [274](#)
  - lava::swapchain, [308](#)
- add\_cmd
  - lava::block, [38](#)
- add\_color\_blend\_attachment
  - lava::render\_pipeline, [263](#)
- add\_command
  - lava::block, [38](#)
- add\_data
  - lava::mesh\_template< T >, [194](#)
- add\_dedicated\_queues
  - lava::device::create\_param, [68](#)
  - queue.hpp, [447](#)
- add\_descriptor
  - lava::pipeline\_layout, [223](#)
- add\_dispatch
  - lava::message\_dispatcher, [202](#)
- add\_dynamic\_state
  - lava::render\_pipeline, [263](#)
- add\_front
  - lava::render\_pass, [253](#)
  - lava::subpass, [295](#)
- add\_id\_map
  - id.hpp, [479](#)
- add\_inactive
  - lava::layer\_list, [188](#)
- add\_mesh
  - lava::producer, [232](#)
- add\_preserve\_attachment
  - lava::subpass, [295](#)
- add\_push\_constant\_range
  - lava::pipeline\_layout, [224](#)
- add\_queue
  - lava::device::create\_param, [69](#)
- add\_queues
  - lava::device::create\_param, [69](#)
  - queue.hpp, [447](#)
- add\_run
  - lava::frame, [129](#)
- add\_run\_end
  - lava::frame, [130](#)
- add\_run\_once
  - lava::frame, [130](#)
- add\_shader
  - lava::render\_pipeline, [263](#)
- add\_shader\_stage
  - lava::render\_pipeline, [263](#)
- add\_specialization\_entry
  - lava::pipeline::shader\_stage, [287](#)
- add\_stage
  - lava::driver, [102](#)
- add\_texture
  - lava::producer, [233](#)
- align
  - data.hpp, [473](#), [474](#)
- align\_up
  - data.hpp, [474](#)
- all
  - lava::app::about\_info\_setting, [15](#)
- alloc
  - lava::device, [85](#)
  - lava::memory, [191](#)
- alloc\_data
  - data.hpp, [474](#)
- allocate
  - lava::data, [72](#)
  - lava::descriptor, [77](#), [78](#)
- allocate\_set
  - lava::descriptor, [78](#)

- allocate\_sets
  - lava::descriptor, [78](#)
- allocator
  - lava::allocator, [16](#)
- app
  - lava::app, [22](#)
- append
  - misc.hpp, [484](#)
- argh.hpp
  - get\_cmd, [536](#)
  - log\_command\_line, [536](#)
- as\_c\_ptr
  - lava::data, [72](#)
- as\_ptr
  - lava::data, [73](#)
- assign
  - lava::window, [333](#)
- attachment
  - lava::attachment, [28](#)
- auto\_bind
  - lava::pipeline, [219](#)
- auto\_line\_width
  - lava::render\_pipeline, [264](#)
- auto\_sizing
  - lava::render\_pipeline, [264](#)
- base.hpp
  - check, [385](#)
  - failed, [385](#)
  - to\_api\_version, [386](#)
  - to\_string, [386](#)
  - to\_version, [386](#)
  - to\_vk\_version, [387](#)
  - vk\_version\_to\_string, [387](#)
- begin
  - misc.hpp, [484](#)
- begin\_frame
  - lava::renderer, [280](#)
- begin\_label
  - debug\_utils.hpp, [392](#)
- benchmark
  - benchmark.hpp, [349](#)
- benchmark.hpp
  - benchmark, [349](#)
  - parse\_benchmark, [349](#)
  - write\_frames\_json, [349](#)
- bind
  - lava::compute\_pipeline, [62](#)
  - lava::mesh\_template< T >, [194](#)
  - lava::pipeline, [219](#)
  - lava::pipeline\_layout, [224](#)
  - lava::render\_pipeline, [264](#)
- bind\_descriptor\_set
  - lava::pipeline\_layout, [224](#)
- bind\_draw
  - lava::mesh\_template< T >, [195](#)
- block\_cmd
  - lava::app, [22](#)
- buffer.hpp
  - buffer\_usage\_to\_possible\_access, [566](#)
  - buffer\_usage\_to\_possible\_stages, [566](#)
- buffer\_usage\_to\_possible\_access
  - buffer.hpp, [566](#)
- buffer\_usage\_to\_possible\_stages
  - buffer.hpp, [566](#)
- busy
  - lava::staging, [291](#)
- c16
  - types.hpp, [503](#)
- c32
  - types.hpp, [503](#)
- c8
  - types.hpp, [503](#)
- c\_data
  - lava::c\_data, [48](#)
- calc\_view\_projection
  - lava::camera, [53](#)
- call
  - lava::device, [85](#)
- capture\_keyboard
  - lava::imgui, [166](#)
- capture\_mouse
  - lava::imgui, [166](#)
- ceil\_div
  - math.hpp, [625](#)
- check
  - base.hpp, [385](#)
  - instance.hpp, [437](#)
  - lava::props, [237](#)
- check\_mod
  - input.hpp, [546](#)
- close\_request
  - lava::window, [333](#)
- collect\_buffers
  - lava::block, [38](#)
- compile\_shader
  - lava::producer, [233](#)
- compute\_queue
  - lava::device, [85](#)
- compute\_queues
  - lava::device, [85](#)
- config.hpp
  - set\_window\_icon, [354](#)
- contains
  - lava::rect, [249](#)
  - misc.hpp, [485](#)
- copy\_from
  - lava::compute\_pipeline, [62](#)
  - lava::render\_pipeline, [264](#)
- copy\_to
  - lava::compute\_pipeline, [62](#)
  - lava::render\_pipeline, [265](#)
- count
  - lava::queue\_family\_info, [244](#)
- create
  - lava::allocator, [17](#)
  - lava::block, [38](#)



- lava::buffer, 44
- lava::camera, 53
- lava::command, 58
- lava::descriptor, 78
- lava::descriptor::pool, 230
- lava::device, 85
- lava::forward\_shading, 126
- lava::id\_registry< T, Meta >, 151
- lava::image, 157
- lava::imgui, 166, 167
- lava::instance, 175
- lava::mesh\_template< T >, 195
- lava::pipeline, 219
- lava::pipeline::shader\_stage, 287
- lava::pipeline\_layout, 224
- lava::platform, 227
- lava::render\_pass, 254
- lava::render\_pipeline, 265
- lava::render\_target, 275
- lava::renderer, 280
- lava::swapchain, 308
- lava::texture, 317
- lava::window, 333
- create\_allocator
  - memory.hpp, 441
- create\_default\_device\_param
  - lava::physical\_device, 212
- create\_default\_texture
  - load\_texture.hpp, 379
- create\_device
  - lava::platform, 227
- create\_folder
  - lava::file\_system, 120
- create\_image
  - image.hpp, 579
- create\_mapped
  - lava::buffer, 44
- create\_mesh
  - lava::producer, 233
  - mesh.hpp, 583
- create\_mesh\_data
  - mesh.hpp, 583
- create\_pipeline\_color\_blend\_attachment
  - render\_pipeline.hpp, 466
- create\_pipeline\_shader\_stage
  - pipeline.hpp, 459
- create\_shader\_module
  - device.hpp, 428
- create\_surface
  - lava::window, 334
  - window.hpp, 560
- create\_target
  - render\_target.hpp, 553
- create\_target\_no\_triple\_buffer
  - render\_target.hpp, 553
- create\_target\_v\_sync
  - render\_target.hpp, 554
- create\_texture
  - lava::producer, 234
- data
  - lava::data, 72
- data.hpp
  - align, 473, 474
  - align\_up, 474
  - alloc\_data, 474
  - free\_data, 475
  - next\_pow\_2, 475
  - realloc\_data, 475
- deallocate
  - lava::descriptor, 79
- deallocate\_set
  - lava::descriptor, 79
- deallocate\_sets
  - lava::descriptor, 79
- debug\_utils.hpp
  - begin\_label, 392
  - end\_label, 393
  - insert\_label, 393
  - set\_acceleration\_structure\_name, 393
  - set\_acceleration\_structure\_nv\_name, 394
  - set\_acceleration\_structure\_nv\_tag, 394
  - set\_acceleration\_structure\_tag, 394
  - set\_buffer\_name, 394
  - set\_buffer\_tag, 395
  - set\_buffer\_view\_name, 395
  - set\_buffer\_view\_tag, 395
  - set\_command\_buffer\_name, 395
  - set\_command\_buffer\_tag, 396
  - set\_command\_pool\_name, 396
  - set\_command\_pool\_tag, 396
  - set\_debug\_report\_callback\_name, 396
  - set\_debug\_report\_callback\_tag, 397
  - set\_debug\_utils\_messenger\_name, 397
  - set\_debug\_utils\_messenger\_tag, 397
  - set\_deferred\_operation\_name, 397
  - set\_deferred\_operation\_tag, 398
  - set\_descriptor\_pool\_name, 398
  - set\_descriptor\_pool\_tag, 398
  - set\_descriptor\_set\_layout\_name, 398
  - set\_descriptor\_set\_layout\_tag, 399
  - set\_descriptor\_set\_name, 399
  - set\_descriptor\_set\_tag, 399
  - set\_descriptor\_update\_template\_name, 399
  - set\_descriptor\_update\_template\_tag, 400
  - set\_device\_memory\_name, 400
  - set\_device\_memory\_tag, 400
  - set\_device\_name, 400
  - set\_device\_tag, 401
  - set\_display\_mode\_name, 401
  - set\_display\_mode\_tag, 401
  - set\_display\_name, 401
  - set\_display\_tag, 402
  - set\_event\_name, 402
  - set\_event\_tag, 402
  - set\_fence\_name, 402
  - set\_fence\_tag, 403

- set\_framebuffer\_name, [403](#)
- set\_framebuffer\_tag, [403](#)
- set\_image\_name, [403](#)
- set\_image\_tag, [404](#)
- set\_image\_view\_name, [404](#)
- set\_image\_view\_tag, [404](#)
- set\_indirect\_commands\_layout\_name, [404](#)
- set\_indirect\_commands\_layout\_tag, [405](#)
- set\_instance\_name, [405](#)
- set\_instance\_tag, [405](#)
- set\_name, [405](#)
- set\_object\_name, [406](#)
- set\_object\_tag, [406](#)
- set\_performance\_configuration\_name, [406](#)
- set\_performance\_configuration\_tag, [407](#)
- set\_physical\_device\_name, [407](#)
- set\_physical\_device\_tag, [407](#)
- set\_pipeline\_cache\_name, [407](#)
- set\_pipeline\_cache\_tag, [408](#)
- set\_pipeline\_layout\_name, [408](#)
- set\_pipeline\_layout\_tag, [408](#)
- set\_pipeline\_name, [408](#)
- set\_pipeline\_tag, [409](#)
- set\_private\_data\_slot\_name, [409](#)
- set\_private\_data\_slot\_tag, [409](#)
- set\_query\_pool\_name, [409](#)
- set\_query\_pool\_tag, [410](#)
- set\_queue\_name, [410](#)
- set\_queue\_tag, [410](#)
- set\_render\_pass\_name, [410](#)
- set\_render\_pass\_tag, [411](#)
- set\_sampler\_name, [411](#)
- set\_sampler\_tag, [411](#)
- set\_sampler\_ycbcr\_conversion\_name, [411](#)
- set\_sampler\_ycbcr\_conversion\_tag, [412](#)
- set\_semaphore\_name, [412](#)
- set\_semaphore\_tag, [412](#)
- set\_shader\_module\_name, [412](#)
- set\_shader\_module\_tag, [413](#)
- set\_surface\_name, [413](#)
- set\_surface\_tag, [413](#)
- set\_swapchain\_name, [413](#)
- set\_swapchain\_tag, [414](#)
- set\_tag, [414](#)
- set\_validation\_cache\_name, [414](#)
- set\_validation\_cache\_tag, [415](#)
- decorated
  - lava::window, [334](#)
- default\_color
  - math.hpp, [626](#)
- default\_queue\_flags
  - queue.hpp, [448](#)
- destroy
  - lava::command, [58](#)
  - lava::image, [157](#)
- detect\_monitor
  - lava::window, [334](#)
- device.hpp
  - create\_shader\_module, [428](#)
  - one\_time\_submit, [429](#)
  - one\_time\_submit\_pool, [429](#)
- draw
  - lava::mesh\_template< T >, [195](#)
- draw\_about
  - lava::app, [22](#)
- driver.hpp
  - LAVA\_STAGE, [538](#)
  - STR, [538](#)
  - STR\_, [538](#)
- elapsed
  - lava::timer, [322](#)
- empty
  - lava::mesh\_template< T >, [195](#)
  - lava::props, [237](#)
- end
  - lava::data, [73](#)
  - misc.hpp, [485](#)
- end\_frame
  - lava::renderer, [280](#)
- end\_label
  - debug\_utils.hpp, [393](#)
- enqueue
  - lava::thread\_pool, [321](#)
- ENUM\_FLAG\_OPERATORS
  - types.hpp, [502](#)
- enumerate\_extension\_properties
  - instance.hpp, [438](#)
- enumerate\_files
  - lava::file\_system, [120](#)
- enumerate\_layer\_properties
  - instance.hpp, [438](#)
- exists
  - lava::file\_system, [120](#)
  - lava::id\_registry< T, Meta >, [151](#)
  - lava::props, [238](#)
  - misc.hpp, [486](#)
- exists\_subpass
  - lava::render\_pass, [254](#)
- extension
  - file\_utils.hpp, [526](#)
- failed
  - base.hpp, [385](#)
- file
  - lava::file, [111](#)
- file.hpp
  - file\_error, [522](#)
- file\_data
  - lava::file\_data, [117](#)
- file\_delete
  - lava::file\_delete, [118](#)
- file\_error
  - file.hpp, [522](#)
- file\_utils.hpp
  - extension, [526](#)
  - get\_filename\_from, [526](#)

- load\_file\_data, 526
- read\_file, 527
- remove\_existing\_path, 527
- write\_file, 527
- find
  - lava::hex\_grid, 143
- find\_memory\_type
  - memory.hpp, 441
- find\_memory\_type\_with\_properties
  - memory.hpp, 441
- find\_supported\_depth\_format
  - format.hpp, 569
- find\_supported\_format
  - format.hpp, 570
- find\_surface\_format
  - format.hpp, 570
- floating
  - lava::window, 334
- flush
  - lava::buffer, 45
- focused
  - lava::window, 334
- format.hpp
  - find\_supported\_depth\_format, 569
  - find\_supported\_format, 570
  - find\_surface\_format, 570
  - format\_align\_dim, 570
  - format\_aspect\_mask, 571
  - format\_bgr, 571
  - format\_block\_dim, 571
  - format\_block\_size, 572
  - format\_depth, 572
  - format\_depth\_stencil, 573
  - format\_num\_blocks, 573
  - format\_srgb, 573
  - format\_stencil, 574
  - image\_memory\_barrier, 574
  - insert\_image\_memory\_barrier, 574
  - set\_image\_layout, 575
  - support\_blit, 576
  - support\_vertex\_buffer\_format, 576
- format\_align\_dim
  - format.hpp, 570
- format\_aspect\_mask
  - format.hpp, 571
- format\_bgr
  - format.hpp, 571
- format\_block\_dim
  - format.hpp, 571
- format\_block\_size
  - format.hpp, 572
- format\_depth
  - format.hpp, 572
- format\_depth\_stencil
  - format.hpp, 573
- format\_num\_blocks
  - format.hpp, 573
- format\_srgb
  - format.hpp, 573
- format\_stencil
  - format.hpp, 574
- format\_string
  - lava::tooltip\_list, 324
- formats
  - lava::surface\_format\_request, 306
- fps\_cap
  - lava::app, 23
- frame
  - lava::frame, 129
  - lava::renderer, 280
- frame.hpp
  - handle\_events, 532
  - handle\_events\_timeout, 532
  - now, 533
- frame\_env
  - lava::frame\_env, 133
- free\_data
  - data.hpp, 475
- fullscreen
  - lava::window, 335
- func
  - lava::input\_callback, 172
- gamepad
  - lava::gamepad, 134
- gamepad.hpp
  - gamepads, 541
- gamepads
  - gamepad.hpp, 541
- get
  - lava::allocator, 17
  - lava::buffer, 45
  - lava::descriptor, 80
  - lava::descriptor::binding, 34
  - lava::descriptor::pool, 230
  - lava::device, 86
  - lava::global\_logger, 139
  - lava::id\_registry< T, Meta >, 151
  - lava::image, 157
  - lava::instance, 175
  - lava::json\_file, 181
  - lava::layer\_list, 188
  - lava::physical\_device, 212
  - lava::pipeline, 219
  - lava::pipeline\_layout, 225
  - lava::pseudorandom\_generator, 242
  - lava::random\_generator, 246
  - lava::render\_pass, 254
  - lava::swapchain, 308
  - lava::window, 335
- get\_address
  - lava::buffer, 45
- get\_all
  - lava::id\_registry< T, Meta >, 152
  - lava::layer\_list, 189
  - lava::props, 238
- get\_all\_meta

- lava::id\_registry< T, Meta >, 152
- get\_allocation
  - lava::buffer, 45
  - lava::image, 157
- get\_allocation\_info
  - lava::buffer, 46
- get\_allocator
  - lava::device, 86
- get\_app
  - lava::file\_system, 120
- get\_aspect\_ratio
  - lava::window, 335
- get\_backbuffer
  - lava::render\_target, 275
- get\_backbuffer\_count
  - lava::swapchain, 309
- get\_backbuffer\_image
  - lava::render\_target, 275
- get\_backbuffers
  - lava::render\_target, 276
  - lava::swapchain, 309
- get\_base\_dir
  - lava::file\_system, 121
- get\_binding\_count
  - lava::descriptor, 80
- get\_bindings
  - lava::descriptor, 80
- get\_clear\_color
  - lava::render\_pass, 254
- get\_clear\_values
  - lava::render\_pass, 255
- get\_cmd
  - argh.hpp, 536
- get\_cmd\_line
  - lava::frame, 130
- get\_cmd\_order
  - lava::block, 39
- get\_color\_space
  - lava::swapchain, 309
- get\_command\_buffer
  - lava::block, 39
- get\_commands
  - lava::block, 39
- get\_compute\_queue
  - lava::device, 86
- get\_compute\_queues
  - lava::device, 86
- get\_content\_scale
  - lava::window, 335
- get\_create\_info
  - lava::pipeline::shader\_stage, 287
- get\_current\_frame
  - lava::block, 40
- get\_current\_time
  - time.hpp, 494
- get\_current\_timestamp
  - time.hpp, 494
- get\_current\_timestamp\_ms
  - time.hpp, 494
- get\_current\_timestamp\_us
  - time.hpp, 495
- get\_data
  - lava::image\_data, 163
  - lava::mesh\_template< T >, 196
- get\_debug\_config
  - lava::instance, 175
- get\_dependency
  - lava::subpass\_dependency, 301
- get\_depth
  - lava::image, 158
- get\_depth\_stencil
  - lava::forward\_shading, 126
- get\_description
  - lava::attachment, 29
  - lava::subpass, 296
- get\_descriptor\_info
  - lava::buffer, 46
  - lava::camera, 53
  - lava::texture, 317
- get\_descriptors
  - lava::pipeline\_layout, 225
- get\_device
  - lava::block, 40
  - lava::buffer, 46
  - lava::descriptor, 80
  - lava::descriptor::pool, 230
  - lava::image, 158
  - lava::pipeline, 220
  - lava::pipeline\_layout, 225
  - lava::render\_pass, 255
  - lava::render\_target, 276
  - lava::renderer, 281
  - lava::swapchain, 309
- get\_device\_memory
  - lava::buffer, 46
- get\_device\_name
  - lava::physical\_device, 212
- get\_device\_type\_string
  - lava::physical\_device, 213
- get\_devices
  - lava::platform, 227
- get\_driver\_version
  - lava::physical\_device, 213
- get\_end\_point
  - lava::rect, 249
- get\_env
  - lava::frame, 130
- get\_ext
  - lava::file\_system, 121
- get\_extension\_properties
  - lava::physical\_device, 213
- get\_features
  - lava::device, 87
  - lava::physical\_device, 213
- get\_filename
  - lava::props, 238

- get\_filename\_from
  - file\_utils.hpp, 526
- get\_first\_physical\_device
  - lava::instance, 176
- get\_format
  - lava::image, 158
  - lava::render\_target, 276
  - lava::swapchain, 309
  - lava::texture, 318
- get\_fps\_info
  - lava::app, 23
- get\_frame
  - lava::renderer, 281
- get\_frame\_count
  - lava::block, 40
  - lava::render\_target, 276
- get\_frame\_counter
  - lava::app, 23
- get\_framebuffer\_size
  - lava::window, 335, 336
- get\_full\_base\_dir
  - lava::file\_system, 121
- get\_graphics\_queue
  - lava::device, 87
- get\_graphics\_queues
  - lava::device, 87
- get\_id
  - lava::entity, 110
  - lava::gamepad, 135
- get\_image
  - lava::render\_target, 276
  - lava::texture, 318
- get\_index\_buffer
  - lava::mesh\_template< T >, 196
- get\_indices
  - lava::mesh\_template< T >, 196
- get\_indices\_count
  - lava::mesh\_template< T >, 197
- get\_info
  - lava::image, 158
  - lava::instance, 176
- get\_ini\_file
  - lava::imgui, 167
- get\_input\_callback
  - lava::imgui, 168
- get\_instance\_version
  - instance.hpp, 438
- get\_json
  - lava::app\_config, 27
  - lava::configurable, 66
  - lava::props, 238
- get\_layout
  - lava::pipeline, 220
- get\_line\_width
  - lava::render\_pipeline, 265
- get\_list
  - lava::id\_listeners< T >, 149
  - lava::tooltip\_list, 324
- get\_mapped\_data
  - lava::buffer, 46
- get\_max
  - lava::descriptor::pool, 230
- get\_memory\_properties
  - lava::physical\_device, 213
- get\_mesh
  - lava::producer, 234
- get\_meta
  - lava::id\_registry< T, Meta >, 152
- get\_mouse\_position
  - lava::input, 171
  - lava::window, 336
- get\_name
  - lava::frame, 131
  - lava::gamepad, 135
- get\_org
  - lava::file\_system, 121
- get\_origin
  - lava::rect, 249
- get\_pad\_id
  - lava::gamepad, 135
- get\_pass
  - lava::forward\_shading, 126
- get\_path
  - lava::file, 112
- get\_physical\_device
  - lava::device, 87
- get\_physical\_devices
  - lava::instance, 176
- get\_pipeline
  - lava::imgui, 168
- get\_position
  - lava::window, 336
- get\_pref\_dir
  - lava::file\_system, 122
- get\_projection
  - lava::camera, 53
- get\_properties
  - lava::device, 88
  - lava::physical\_device, 214
- get\_push\_constant\_ranges
  - lava::pipeline\_layout, 225
- get\_queue\_family
  - lava::physical\_device, 214
- get\_queue\_family\_properties
  - lava::physical\_device, 214
- get\_queues
  - lava::device, 88
- get\_real\_dir
  - lava::file\_system, 122
- get\_render\_pass
  - lava::render\_pipeline, 265
- get\_res\_dir
  - lava::file\_system, 122
- get\_running\_time
  - lava::frame, 131
- get\_running\_time\_sec

- lava::frame, 131
- get\_save\_name
  - lava::window, 337
- get\_scancode
  - input.hpp, 546
- get\_scissor
  - lava::render\_pipeline, 265
- get\_shader
  - lava::producer, 234
- get\_shader\_stage
  - lava::compute\_pipeline, 62
- get\_shader\_stages
  - lava::render\_pipeline, 266
- get\_size
  - lava::buffer, 47
  - lava::file, 112
  - lava::image, 158
  - lava::rect, 249
  - lava::render\_target, 277
  - lava::swapchain, 310
  - lava::texture, 318
  - lava::window, 337
- get\_sizes
  - lava::descriptor::pool, 231
- get\_sizing
  - lava::render\_pipeline, 266
- get\_stages
  - lava::driver, 103
- get\_state
  - lava::window, 337
- get\_subpass
  - lava::render\_pass, 255
  - lava::render\_pipeline, 266
- get\_subpass\_count
  - lava::render\_pass, 255
- get\_subpasses
  - lava::render\_pass, 256
- get\_subresource\_range
  - lava::image, 159
- get\_swapchain
  - lava::render\_target, 277
- get\_target\_callback
  - lava::render\_pass, 256
- get\_texture
  - lava::producer, 235
- get\_title
  - lava::window, 337
- get\_transfer\_queue
  - lava::device, 88
- get\_transfer\_queues
  - lava::device, 88
- get\_type
  - lava::file, 112
  - lava::texture, 318
- get\_version
  - lava::file\_system, 122
- get\_vertex\_buffer
  - lava::mesh\_template< T >, 197
- get\_vertices
  - lava::mesh\_template< T >, 197
- get\_vertices\_count
  - lava::mesh\_template< T >, 198
- get\_view
  - lava::camera, 54
  - lava::image, 159
- get\_view\_info
  - lava::image, 159
- get\_viewport
  - lava::render\_pipeline, 266
- get\_vk\_pass
  - lava::forward\_shading, 126
- get\_vk\_physical\_device
  - lava::device, 89
- get\_window
  - window.hpp, 560
- grab\_image
  - image.hpp, 579
- graphics\_queue
  - lava::device, 89
- graphics\_queues
  - lava::device, 89
- handle
  - lava::camera, 54
- handle\_events
  - frame.hpp, 532
- handle\_events\_timeout
  - frame.hpp, 532
- handle\_resize
  - lava::window, 338
- has\_dispatch
  - lava::message\_dispatcher, 202
- hash256
  - math.hpp, 625
- hash\_combine
  - types.hpp, 505
- hash\_value
  - types.hpp, 506
- hex.hpp
  - hex\_calculate\_inner\_radius, 600
  - hex\_cardinal\_directions, 610
  - hex\_cell\_from\_pair, 600
  - hex\_corner\_offset, 601
  - hex\_diagonal, 601
  - hex\_diagonal\_neighbor, 601
  - hex\_diagonals, 610
  - hex\_direction, 602
  - hex\_directions, 610
  - hex\_distance, 602
  - hex\_get, 602
  - hex\_get\_corner, 603
  - hex\_get\_s, 603
  - hex\_is\_valid, 603
  - hex\_layout\_flat, 610
  - hex\_layout\_point\_y, 611
  - hex\_length, 604
  - hex\_lerp, 604

- hex\_line, 604
- hex\_neighbor, 605
- hex\_opposite, 605
- hex\_pixel\_to\_cell, 605
- hex\_polygon\_corners, 606
- hex\_q\_doubled\_from\_cube, 606
- hex\_q\_doubled\_to\_cube, 606
- hex\_q\_offset\_from\_cube, 607
- hex\_q\_offset\_to\_cube, 607
- hex\_r\_doubled\_from\_cube, 607
- hex\_r\_doubled\_to\_cube, 608
- hex\_r\_offset\_from\_cube, 608
- hex\_r\_offset\_to\_cube, 608
- hex\_round, 609
- hex\_to\_pixel, 609
- to\_string, 609
- hex\_calculate\_inner\_radius
  - hex.hpp, 600
- hex\_cardinal\_directions
  - hex.hpp, 610
- hex\_cell\_from\_pair
  - hex.hpp, 600
- hex\_corner\_offset
  - hex.hpp, 601
- hex\_diagonal
  - hex.hpp, 601
- hex\_diagonal\_neighbor
  - hex.hpp, 601
- hex\_diagonals
  - hex.hpp, 610
- hex\_direction
  - hex.hpp, 602
- hex\_directions
  - hex.hpp, 610
- hex\_distance
  - hex.hpp, 602
- hex\_get
  - hex.hpp, 602
- hex\_get\_corner
  - hex.hpp, 603
- hex\_get\_s
  - hex.hpp, 603
- hex\_grid
  - lava::hex\_grid, 142
- hex\_is\_valid
  - hex.hpp, 603
- hex\_layout\_flat
  - hex.hpp, 610
- hex\_layout\_point\_y
  - hex.hpp, 611
- hex\_length
  - hex.hpp, 604
- hex\_lerp
  - hex.hpp, 604
- hex\_line
  - hex.hpp, 604
- hex\_neighbor
  - hex.hpp, 605
- hex\_opposite
  - hex.hpp, 605
- hex\_pixel\_to\_cell
  - hex.hpp, 605
- hex\_polygon\_corners
  - hex.hpp, 606
- hex\_q\_doubled\_from\_cube
  - hex.hpp, 606
- hex\_q\_doubled\_to\_cube
  - hex.hpp, 606
- hex\_q\_offset\_from\_cube
  - hex.hpp, 607
- hex\_q\_offset\_to\_cube
  - hex.hpp, 607
- hex\_r\_doubled\_from\_cube
  - hex.hpp, 607
- hex\_r\_doubled\_to\_cube
  - hex.hpp, 608
- hex\_r\_offset\_from\_cube
  - hex.hpp, 608
- hex\_r\_offset\_to\_cube
  - hex.hpp, 608
- hex\_round
  - hex.hpp, 609
- hex\_to\_pixel
  - hex.hpp, 609
- hovered
  - lava::window, 338
- i16
  - types.hpp, 503
- i32
  - types.hpp, 503
- i64
  - types.hpp, 504
- i8
  - types.hpp, 504
- iconified
  - lava::window, 338
- id
  - lava::id, 147
- id.hpp
  - add\_id\_map, 479
  - remove\_id\_map, 479
  - to\_id, 480
- image
  - lava::image, 156
- image.hpp
  - create\_image, 579
  - grab\_image, 579
- image\_memory\_barrier
  - format.hpp, 574
- imgui
  - lava::imgui, 166
- imgui.hpp
  - imgui\_left\_spacing, 371
  - setup\_imgui\_font, 372
  - setup\_imgui\_font\_icons, 372
- imgui\_left\_spacing

- imgui.hpp, 371
- initialize
  - lava::file\_system, 123
  - lava::physical\_device, 214
- input.hpp
  - check\_mod, 546
  - get\_scancode, 546
  - to\_string, 546
- insert\_image\_memory\_barrier
  - format.hpp, 574
- insert\_label
  - debug\_utils.hpp, 393
- install
  - lava::props, 239
- instance
  - lava::driver, 103
  - lava::ids, 154
  - lava::memory, 191
- instance.hpp
  - check, 437
  - enumerate\_extension\_properties, 438
  - enumerate\_layer\_properties, 438
  - get\_instance\_version, 438
- item
  - lava::props::item, 179
- json\_file
  - lava::json\_file, 180
- lava::allocator, 16
  - allocator, 16
  - create, 17
  - get, 17
  - make, 17
  - valid, 17
- lava::app, 18
  - app, 22
  - block\_cmd, 22
  - draw\_about, 22
  - fps\_cap, 23
  - get\_fps\_info, 23
  - get\_frame\_counter, 23
  - screenshot, 23
  - setup, 23
  - switch\_config, 24
  - triple\_buffer, 25
  - v\_sync, 25
- lava::app::about\_info\_setting, 15
  - all, 15
- lava::app\_config, 25
  - get\_json, 27
  - set\_json, 27
- lava::attachment, 27
  - attachment, 28
  - get\_description, 29
  - make, 29
  - set\_final\_layout, 29
  - set\_format, 29
  - set\_initial\_layout, 30
  - set\_layouts, 30
  - set\_load\_op, 30
  - set\_op, 30
  - set\_samples, 31
  - set\_stencil\_load\_op, 31
  - set\_stencil\_op, 31
  - set\_stencil\_store\_op, 31
  - set\_store\_op, 32
- lava::benchmark\_data, 32
- lava::block, 36
  - activated, 37
  - add\_cmd, 38
  - add\_command, 38
  - collect\_buffers, 38
  - create, 38
  - get\_cmd\_order, 39
  - get\_command\_buffer, 39
  - get\_commands, 39
  - get\_current\_frame, 40
  - get\_device, 40
  - get\_frame\_count, 40
  - make, 40
  - process, 40
  - remove\_cmd, 41
  - remove\_command, 41
  - set\_active, 41
- lava::buffer, 42
  - create, 44
  - create\_mapped, 44
  - flush, 45
  - get, 45
  - get\_address, 45
  - get\_allocation, 45
  - get\_allocation\_info, 46
  - get\_descriptor\_info, 46
  - get\_device, 46
  - get\_device\_memory, 46
  - get\_mapped\_data, 46
  - get\_size, 47
  - make, 47
  - valid, 47
- lava::c\_data, 47
  - c\_data, 48
- lava::camera, 50
  - activated, 53
  - calc\_view\_projection, 53
  - create, 53
  - get\_descriptor\_info, 53
  - get\_projection, 53
  - get\_view, 54
  - handle, 54
  - moving, 55
  - set\_active, 55
  - set\_movement\_keys, 55
  - update\_view, 55, 56
  - valid, 56
- lava::command, 56
  - create, 58



- destroy, 58
- make, 58
- lava::compute\_pipeline, 59
  - bind, 62
  - copy\_from, 62
  - copy\_to, 62
  - get\_shader\_stage, 62
  - make, 62
  - set, 64
  - set\_shader\_stage, 64
- lava::configurable, 66
  - get\_json, 66
  - set\_json, 66
- lava::data, 71
  - allocate, 72
  - as\_c\_ptr, 72
  - as\_ptr, 73
  - data, 72
  - end, 73
  - set, 73
- lava::data\_provider, 74
- lava::descriptor, 75
  - add, 77
  - add\_binding, 77
  - allocate, 77, 78
  - allocate\_set, 78
  - allocate\_sets, 78
  - create, 78
  - deallocate, 79
  - deallocate\_set, 79
  - deallocate\_sets, 79
  - get, 80
  - get\_binding\_count, 80
  - get\_bindings, 80
  - get\_device, 80
  - make, 80
- lava::descriptor::binding, 33
  - get, 34
  - make, 34
  - set, 34
  - set\_count, 34
  - set\_samplers, 35
  - set\_stage\_flags, 35
  - set\_type, 35
- lava::descriptor::pool, 228
  - create, 230
  - get, 230
  - get\_device, 230
  - get\_max, 230
  - get\_sizes, 231
  - make, 231
- lava::device, 81
  - alloc, 85
  - call, 85
  - compute\_queue, 85
  - compute\_queues, 85
  - create, 85
  - get, 86
  - get\_allocator, 86
  - get\_compute\_queue, 86
  - get\_compute\_queues, 86
  - get\_features, 87
  - get\_graphics\_queue, 87
  - get\_graphics\_queues, 87
  - get\_physical\_device, 87
  - get\_properties, 88
  - get\_queues, 88
  - get\_transfer\_queue, 88
  - get\_transfer\_queues, 88
  - get\_vk\_physical\_device, 89
  - graphics\_queue, 89
  - graphics\_queues, 89
  - make, 89
  - queues, 89
  - set\_allocator, 89
  - surface\_supported, 90
  - transfer\_queue, 90
  - transfer\_queues, 90
  - wait\_for\_idle, 90
- lava::device::create\_param, 67
  - add\_dedicated\_queues, 68
  - add\_queue, 69
  - add\_queues, 69
  - verify\_queues, 69
- lava::device\_table, 91
  - vkAcquireNextImageKHR, 92
  - vkAllocateCommandBuffers, 92, 93
  - vkCreateCommandPool, 93
  - vkCreateFence, 94
  - vkCreateImageView, 94
  - vkCreateSampler, 95
  - vkCreateSemaphore, 95
  - vkCreateShaderModule, 96
  - vkCreateSwapchainKHR, 96
  - vkDestroyCommandPool, 97
  - vkDestroyFence, 97
  - vkDestroyImageView, 97
  - vkDestroySampler, 97
  - vkDestroySemaphore, 98
  - vkDestroySwapchainKHR, 98
  - vkFreeCommandBuffers, 98
  - vkGetSwapchainImagesKHR, 98
  - vkQueuePresentKHR, 99
  - vkQueueSubmit, 99
  - vkResetFences, 99
  - vkUpdateDescriptorSets, 99–101
  - vkWaitForFences, 101
- lava::driver, 102
  - add\_stage, 102
  - get\_stages, 103
  - instance, 103
  - run, 103
- lava::driver::result, 281
- lava::engine, 104
  - setup, 108
- lava::entity, 108

- get\_id, 110
- lava::file, 110
  - file, 111
  - get\_path, 112
  - get\_size, 112
  - get\_type, 112
  - open, 112
  - opened, 113
  - read, 113
  - seek, 113
  - tell, 114
  - writable, 114
  - write, 114
- lava::file\_data, 115
  - file\_data, 117
- lava::file\_delete, 117
  - file\_delete, 118
- lava::file\_system, 118
  - create\_folder, 120
  - enumerate\_files, 120
  - exists, 120
  - get\_app, 120
  - get\_base\_dir, 121
  - get\_ext, 121
  - get\_full\_base\_dir, 121
  - get\_org, 121
  - get\_pref\_dir, 122
  - get\_real\_dir, 122
  - get\_res\_dir, 122
  - get\_version, 122
  - initialize, 123
  - mount, 123
  - mount\_base, 123
  - mount\_res, 124
  - ready, 124
- lava::forward\_shading, 125
  - create, 126
  - get\_depth\_stencil, 126
  - get\_pass, 126
  - get\_vk\_pass, 126
- lava::frame, 127
  - add\_run, 129
  - add\_run\_end, 130
  - add\_run\_once, 130
  - frame, 129
  - get\_cmd\_line, 130
  - get\_env, 130
  - get\_name, 131
  - get\_running\_time, 131
  - get\_running\_time\_sec, 131
  - ready, 131
  - remove, 131
  - run, 132
  - set\_wait\_for\_events, 132
  - shut\_down, 132
  - waiting\_for\_events, 132
- lava::frame\_env, 133
  - frame\_env, 133
- lava::gamepad, 134
  - gamepad, 134
  - get\_id, 135
  - get\_name, 135
  - get\_pad\_id, 135
  - pressed, 135
  - ready, 136
  - update, 136
  - value, 136
- lava::gamepad\_manager, 137
  - add, 137
  - remove, 137
  - singleton, 138
- lava::global\_logger, 138
  - get, 139
  - set, 139
  - singleton, 139
- lava::hex\_cell, 139
  - add, 140
  - scale, 141
  - subtract, 141
  - to\_pair, 141
- lava::hex\_fractional\_cell, 141
- lava::hex\_grid, 142
  - find, 143
  - hex\_grid, 142
  - to\_pixel, 143
  - update, 143
- lava::hex\_layout, 144
- lava::hex\_offset\_coord, 144
- lava::hex\_orientation, 145
- lava::hex\_point, 145
- lava::id, 147
  - id, 147
  - to\_string, 148
  - valid, 148
- lava::id\_listeners< T >, 148
  - add, 149
  - get\_list, 149
  - remove, 149
- lava::id\_registry< T, Meta >, 150
  - add, 151
  - create, 151
  - exists, 151
  - get, 151
  - get\_all, 152
  - get\_all\_meta, 152
  - get\_meta, 152
  - remove, 152
  - update, 153
- lava::ids, 153
  - instance, 154
  - next, 154
- lava::image, 154
  - create, 157
  - destroy, 157
  - get, 157
  - get\_allocation, 157

- get\_depth, 158
- get\_device, 158
- get\_format, 158
- get\_info, 158
- get\_size, 158
- get\_subresource\_range, 159
- get\_view, 159
- get\_view\_info, 159
- image, 156
- make, 159
- set\_aspect\_mask, 160
- set\_component, 160
- set\_flags, 160
- set\_layer\_count, 160
- set\_layout, 161
- set\_level\_count, 161
- set\_tiling, 161
- set\_usage, 161
- set\_view\_type, 162
- lava::image\_data, 162
  - get\_data, 163
  - ready, 163
  - set\_data, 163
  - size, 163
- lava::imgui, 164
  - activated, 166
  - capture\_keyboard, 166
  - capture\_mouse, 166
  - create, 166, 167
  - get\_ini\_file, 167
  - get\_input\_callback, 168
  - get\_pipeline, 168
  - imgui, 166
  - ready, 168
  - set\_active, 168
  - set\_ini\_file, 169
  - setup, 169
  - upload\_fonts, 169
- lava::imgui::config, 64
- lava::imgui::font, 124
- lava::imgui::icon\_font, 146
- lava::input, 170
  - add, 171
  - get\_mouse\_position, 171
  - remove, 171
  - set\_mouse\_position, 171
- lava::input\_callback, 172
  - func, 172
- lava::input\_events< T >, 173
  - add, 173
- lava::instance, 174
  - create, 175
  - get, 175
  - get\_debug\_config, 175
  - get\_first\_physical\_device, 176
  - get\_info, 176
  - get\_physical\_devices, 176
  - singleton, 176
- lava::instance::create\_param, 70
- lava::instance::debug\_config, 74
- lava::instance\_info, 177
- lava::interface, 177
- lava::json\_file, 179
  - add, 180
  - get, 181
  - json\_file, 180
  - load, 181
  - remove, 181
  - save, 181
  - set, 181
- lava::json\_file::callback, 49
- lava::key\_event, 182
  - active, 183
  - pressed, 183
  - released, 183
  - repeated, 184
- lava::layer, 184
  - layer, 186
  - make, 186
- lava::layer\_list, 187
  - add, 188
  - add\_inactive, 188
  - get, 188
  - get\_all, 189
  - remove, 189
- lava::log::config, 65
- lava::memory, 190
  - alloc, 191
  - instance, 191
  - set\_callbacks, 191
  - set\_use\_custom\_cpu\_callbacks, 191
- lava::mesh\_meta, 192
- lava::mesh\_template< T >, 192
  - add\_data, 194
  - bind, 194
  - bind\_draw, 195
  - create, 195
  - draw, 195
  - empty, 195
  - get\_data, 196
  - get\_index\_buffer, 196
  - get\_indices, 196
  - get\_indices\_count, 197
  - get\_vertex\_buffer, 197
  - get\_vertices, 197
  - get\_vertices\_count, 198
  - make, 198
  - reload, 198
  - set\_data, 198
- lava::mesh\_template\_data< T >, 199
  - move, 199
  - scale, 200
  - scale\_vector, 200
- lava::message\_dispatcher, 201
  - add\_dispatch, 202
  - has\_dispatch, 202

- remove\_dispatch, 202
- send\_message, 202
- setup, 203
- update, 203
- lava::mouse\_active\_event, 204
- lava::mouse\_button\_event, 205
  - pressed, 205
  - released, 206
- lava::mouse\_move\_event, 206
- lava::mouse\_position, 207
- lava::no\_copy\_no\_move, 208
- lava::pair\_hash, 209
  - operator(), 209
- lava::path\_drop\_event, 209
- lava::physical\_device, 210
  - create\_default\_device\_param, 212
  - get, 212
  - get\_device\_name, 212
  - get\_device\_type\_string, 213
  - get\_driver\_version, 213
  - get\_extension\_properties, 213
  - get\_features, 213
  - get\_memory\_properties, 213
  - get\_properties, 214
  - get\_queue\_family, 214
  - get\_queue\_family\_properties, 214
  - initialize, 214
  - make, 215
  - physical\_device, 212
  - supported, 215
  - surface\_supported, 215
  - swapchain\_supported, 215
- lava::pipeline, 216
  - activated, 219
  - auto\_bind, 219
  - bind, 219
  - create, 219
  - get, 219
  - get\_device, 220
  - get\_layout, 220
  - pipeline, 218
  - ready, 220
  - set\_active, 220
  - set\_auto\_bind, 221
  - set\_layout, 221
  - setup, 221
- lava::pipeline::shader\_stage, 286
  - add\_specialization\_entry, 287
  - create, 287
  - get\_create\_info, 287
  - make, 287
  - set\_stage, 289
- lava::pipeline\_layout, 222
  - add, 223
  - add\_descriptor, 223
  - add\_push\_constant\_range, 224
  - bind, 224
  - bind\_descriptor\_set, 224
  - create, 224
  - get, 225
  - get\_descriptors, 225
  - get\_device, 225
  - get\_push\_constant\_ranges, 225
  - make, 225
- lava::platform, 226
  - create, 227
  - create\_device, 227
  - get\_devices, 227
  - remove, 228
- lava::producer, 231
  - add\_mesh, 232
  - add\_texture, 233
  - compile\_shader, 233
  - create\_mesh, 233
  - create\_texture, 234
  - get\_mesh, 234
  - get\_shader, 234
  - get\_texture, 235
  - reload\_shader, 235
- lava::props, 236
  - add, 237
  - check, 237
  - empty, 237
  - exists, 238
  - get\_all, 238
  - get\_filename, 238
  - get\_json, 238
  - install, 239
  - load, 239
  - load\_all, 239
  - operator(), 239
  - parse, 240
  - remove, 240
  - set\_filename, 240
  - set\_json, 240
  - unload, 241
- lava::props::item, 178
  - item, 179
- lava::pseudorandom\_generator, 241
  - get, 242
  - pseudorandom\_generator, 241
  - set\_seed, 242
- lava::queue, 242
  - operator<, 243
  - valid, 243
- lava::queue\_family\_info, 244
  - add, 244
  - count, 244
- lava::queue\_info, 245
- lava::random\_generator, 245
  - get, 246
- lava::rect, 247
  - contains, 249
  - get\_end\_point, 249
  - get\_origin, 249
  - get\_size, 249

- move, [249](#)
- rect, [248](#)
- set\_size, [250](#)
- lava::render\_pass, [250](#)
  - add, [252](#), [253](#)
  - add\_front, [253](#)
  - create, [254](#)
  - exists\_subpass, [254](#)
  - get, [254](#)
  - get\_clear\_color, [254](#)
  - get\_clear\_values, [255](#)
  - get\_device, [255](#)
  - get\_subpass, [255](#)
  - get\_subpass\_count, [255](#)
  - get\_subpasses, [256](#)
  - get\_target\_callback, [256](#)
  - make, [256](#)
  - process, [256](#)
  - remove, [257](#)
  - render\_pass, [252](#)
  - set\_clear\_color, [257](#)
  - set\_clear\_values, [257](#)
- lava::render\_pipeline, [258](#)
  - add, [262](#)
  - add\_color\_blend\_attachment, [263](#)
  - add\_dynamic\_state, [263](#)
  - add\_shader, [263](#)
  - add\_shader\_stage, [263](#)
  - auto\_line\_width, [264](#)
  - auto\_sizing, [264](#)
  - bind, [264](#)
  - copy\_from, [264](#)
  - copy\_to, [265](#)
  - create, [265](#)
  - get\_line\_width, [265](#)
  - get\_render\_pass, [265](#)
  - get\_scissor, [265](#)
  - get\_shader\_stages, [266](#)
  - get\_sizing, [266](#)
  - get\_subpass, [266](#)
  - get\_viewport, [266](#)
  - make, [266](#)
  - render\_pipeline, [262](#)
  - set, [267](#)
  - set\_auto\_line\_width, [267](#)
  - set\_auto\_size, [267](#)
  - set\_depth\_compare\_op, [267](#)
  - set\_depth\_test\_and\_write, [268](#)
  - set\_dynamic\_states, [268](#)
  - set\_input\_topology, [268](#)
  - set\_line\_width, [268](#), [269](#)
  - set\_rasterization\_cull\_mode, [269](#)
  - set\_rasterization\_front\_face, [269](#)
  - set\_rasterization\_polygon\_mode, [269](#)
  - set\_render\_pass, [270](#)
  - set\_scissor, [270](#)
  - set\_sizing, [270](#)
  - set\_subpass, [270](#)
  - set\_vertex\_input\_attribute, [271](#)
  - set\_vertex\_input\_attributes, [271](#)
  - set\_vertex\_input\_binding, [271](#)
  - set\_vertex\_input\_bindings, [271](#)
  - set\_viewport, [272](#)
  - set\_viewport\_and\_scissor, [272](#)
- lava::render\_pipeline::create\_info, [67](#)
- lava::render\_target, [272](#)
  - add\_callback, [274](#)
  - create, [275](#)
  - get\_backbuffer, [275](#)
  - get\_backbuffer\_image, [275](#)
  - get\_backbuffers, [276](#)
  - get\_device, [276](#)
  - get\_format, [276](#)
  - get\_frame\_count, [276](#)
  - get\_image, [276](#)
  - get\_size, [277](#)
  - get\_swapchain, [277](#)
  - make, [277](#)
  - reload\_request, [277](#)
  - remove\_callback, [277](#)
  - resize, [278](#)
- lava::renderer, [278](#)
  - begin\_frame, [280](#)
  - create, [280](#)
  - end\_frame, [280](#)
  - frame, [280](#)
  - get\_device, [281](#)
  - get\_frame, [281](#)
- lava::reversion\_wrapper< T >, [282](#)
- lava::run\_time, [282](#)
- lava::scoped\_label< T >, [283](#)
  - scoped\_label, [284](#)
- lava::scroll\_event, [284](#)
- lava::scroll\_offset, [285](#)
- lava::semantic\_version, [285](#)
- lava::stage, [289](#)
  - stage, [290](#)
- lava::staging, [290](#)
  - add, [291](#)
  - busy, [291](#)
  - stage, [291](#)
- lava::subpass, [293](#)
  - activated, [295](#)
  - add, [295](#)
  - add\_front, [295](#)
  - add\_preserve\_attachment, [295](#)
  - get\_description, [296](#)
  - make, [296](#)
  - process, [296](#)
  - remove, [296](#)
  - set, [297](#)
  - set\_active, [297](#)
  - set\_color\_attachment, [297](#)
  - set\_color\_attachments, [298](#)
  - set\_depth\_stencil\_attachment, [298](#)
  - set\_input\_attachment, [298](#), [299](#)

- set\_input\_attachments, 299
- set\_preserve\_attachments, 299
- set\_resolve\_attachment, 299, 300
- set\_resolve\_attachments, 300
- lava::subpass\_dependency, 300
  - get\_dependency, 301
  - make, 301
  - set\_access\_mask, 303
  - set\_dependency\_flags, 303
  - set\_dst\_access\_mask, 303
  - set\_dst\_stage\_mask, 303
  - set\_dst\_subpass, 304
  - set\_src\_access\_mask, 304
  - set\_src\_stage\_mask, 304
  - set\_src\_subpass, 304
  - set\_stage\_mask, 305
  - set\_subpass, 305
- lava::surface\_format\_request, 305
  - formats, 306
- lava::swapchain, 306
  - add\_callback, 308
  - create, 308
  - get, 308
  - get\_backbuffer\_count, 309
  - get\_backbuffers, 309
  - get\_color\_space, 309
  - get\_device, 309
  - get\_format, 309
  - get\_size, 310
  - reload\_request, 310
  - remove\_callback, 310
  - resize, 310
  - surface\_supported, 311
  - triple\_buffer, 311
  - v\_sync, 311
- lava::swapchain::callback, 49
- lava::target\_callback, 312
- lava::telegram, 312
  - operator<, 313
  - operator==, 314
  - telegram, 313
- lava::telegraph, 314
  - send\_message, 315
- lava::texture, 315
  - create, 317
  - get\_descriptor\_info, 317
  - get\_format, 318
  - get\_image, 318
  - get\_size, 318
  - get\_type, 318
  - make, 318
  - stage, 319
  - upload, 319
- lava::texture::layer, 186
- lava::texture::mip\_level, 203
- lava::texture\_file, 320
- lava::thread\_pool, 320
  - enqueue, 321
  - setup, 321
- lava::timer, 321
  - elapsed, 322
- lava::tooltip, 322
  - tooltip, 323
- lava::tooltip\_list, 323
  - add, 324
  - format\_string, 324
  - get\_list, 324
  - set, 324
- lava::u\_data, 325
  - u\_data, 326
- lava::version, 327
- lava::vertex, 327
  - operator==, 328
- lava::vk\_result, 328
  - operator bool, 329
- lava::window, 329
  - assign, 333
  - close\_request, 333
  - create, 333
  - create\_surface, 334
  - decorated, 334
  - detect\_monitor, 334
  - floating, 334
  - focused, 334
  - fullscreen, 335
  - get, 335
  - get\_aspect\_ratio, 335
  - get\_content\_scale, 335
  - get\_framebuffer\_size, 335, 336
  - get\_mouse\_position, 336
  - get\_position, 336
  - get\_save\_name, 337
  - get\_size, 337
  - get\_state, 337
  - get\_title, 337
  - handle\_resize, 338
  - hovered, 338
  - iconified, 338
  - maximized, 338
  - resizable, 338
  - resize\_request, 339
  - save\_title, 339
  - set\_decorated, 339
  - set\_floating, 339
  - set\_fullscreen, 340
  - set\_icon, 340
  - set\_mouse\_position, 340
  - set\_position, 340
  - set\_resizable, 341
  - set\_save\_name, 341
  - set\_size, 341
  - set\_state, 341
  - set\_title, 342
  - show\_save\_title, 342
  - switch\_mode, 342
  - switch\_mode\_request, 342

- visible, [342](#)
- window, [333](#)
- lava::window::state, [292](#)
- LAVA\_STAGE
  - driver.hpp, [538](#)
- layer
  - lava::layer, [186](#)
- liblava/app.hpp, [345](#)
- liblava/app/app.hpp, [346](#)
- liblava/app/benchmark.hpp, [348](#), [351](#)
- liblava/app/camera.hpp, [351](#), [352](#)
- liblava/app/config.hpp, [354](#), [355](#)
- liblava/app/forward\_shading.hpp, [355](#), [356](#)
- liblava/app/icon.hpp, [356](#), [357](#)
- liblava/app/imgui.hpp, [370](#), [372](#)
- liblava/asset.hpp, [375](#)
- liblava/asset/load\_image.hpp, [376](#), [377](#)
- liblava/asset/load\_mesh.hpp, [377](#), [378](#)
- liblava/asset/load\_texture.hpp, [378](#), [380](#)
- liblava/asset/write\_image.hpp, [380](#), [381](#)
- liblava/base.hpp, [381](#), [382](#)
- liblava/base/base.hpp, [382](#), [388](#)
- liblava/base/debug\_utils.hpp, [389](#), [415](#)
- liblava/base/device.hpp, [428](#), [430](#)
- liblava/base/device\_table.hpp, [432](#), [433](#)
- liblava/base/instance.hpp, [437](#), [439](#)
- liblava/base/memory.hpp, [440](#), [442](#)
- liblava/base/physical\_device.hpp, [443](#), [444](#)
- liblava/base/platform.hpp, [445](#)
- liblava/base/queue.hpp, [446](#), [449](#)
- liblava/block.hpp, [450](#)
- liblava/block/attachment.hpp, [453](#)
- liblava/block/block.hpp, [451](#)
- liblava/block/compute\_pipeline.hpp, [454](#), [455](#)
- liblava/block/descriptor.hpp, [456](#)
- liblava/block/pipeline.hpp, [458](#), [460](#)
- liblava/block/pipeline\_layout.hpp, [461](#), [462](#)
- liblava/block/render\_pass.hpp, [463](#), [464](#)
- liblava/block/render\_pipeline.hpp, [465](#), [466](#)
- liblava/block/subpass.hpp, [469](#), [470](#)
- liblava/core.hpp, [472](#)
- liblava/core/data.hpp, [472](#), [476](#)
- liblava/core/id.hpp, [478](#), [480](#)
- liblava/core/misc.hpp, [482](#), [491](#)
- liblava/core/time.hpp, [492](#), [497](#)
- liblava/core/types.hpp, [499](#), [510](#)
- liblava/core/version.hpp, [513](#), [514](#)
- liblava/engine.hpp, [515](#)
- liblava/engine/engine.hpp, [516](#)
- liblava/engine/producer.hpp, [517](#), [518](#)
- liblava/engine/props.hpp, [519](#)
- liblava/file.hpp, [520](#), [521](#)
- liblava/file/file.hpp, [521](#), [522](#)
- liblava/file/file\_system.hpp, [523](#), [524](#)
- liblava/file/file\_utils.hpp, [525](#), [528](#)
- liblava/file/json.hpp, [528](#)
- liblava/file/json\_file.hpp, [529](#)
- liblava/frame.hpp, [530](#), [531](#)
- liblava/frame/argh.hpp, [535](#), [537](#)
- liblava/frame/driver.hpp, [537](#), [539](#)
- liblava/frame/frame.hpp, [531](#), [533](#)
- liblava/frame/gamepad.hpp, [540](#), [541](#)
- liblava/frame/input.hpp, [543](#), [547](#)
- liblava/frame/render\_target.hpp, [552](#), [554](#)
- liblava/frame/renderer.hpp, [556](#), [557](#)
- liblava/frame/swapchain.hpp, [557](#), [558](#)
- liblava/frame/window.hpp, [559](#), [561](#)
- liblava/lava.hpp, [564](#)
- liblava/resource.hpp, [565](#)
- liblava/resource/buffer.hpp, [565](#), [567](#)
- liblava/resource/format.hpp, [568](#), [577](#)
- liblava/resource/image.hpp, [578](#), [579](#)
- liblava/resource/mesh.hpp, [581](#), [585](#)
- liblava/resource/primitive.hpp, [592](#), [593](#)
- liblava/resource/texture.hpp, [594](#), [595](#)
- liblava/test.hpp, [596](#), [597](#)
- liblava/util.hpp, [597](#)
- liblava/util/hex.hpp, [597](#), [611](#)
- liblava/util/layer.hpp, [616](#), [617](#)
- liblava/util/log.hpp, [618](#), [622](#)
- liblava/util/math.hpp, [624](#), [626](#)
- liblava/util/random.hpp, [628](#), [629](#)
- liblava/util/telegram.hpp, [630](#), [631](#)
- liblava/util/thread.hpp, [633](#), [634](#)
- load
  - lava::json\_file, [181](#)
  - lava::props, [239](#)
- load\_all
  - lava::props, [239](#)
- load\_file\_data
  - file\_utils.hpp, [526](#)
- load\_image
  - load\_image.hpp, [376](#)
- load\_image.hpp
  - load\_image, [376](#)
- load\_mesh
  - load\_mesh.hpp, [377](#)
- load\_mesh.hpp
  - load\_mesh, [377](#)
- load\_texture
  - load\_texture.hpp, [379](#)
- load\_texture.hpp
  - create\_default\_texture, [379](#)
  - load\_texture, [379](#)
- log.hpp
  - logger, [619](#)
  - sem\_version\_string, [619](#)
  - semantic\_version\_string, [620](#)
  - setup, [620](#)
  - teardown, [620](#)
  - to\_string, [620](#), [621](#)
  - version\_string, [622](#)
- log\_command\_line
  - argh.hpp, [536](#)
- logger
  - log.hpp, [619](#)

- make
  - lava::allocator, 17
  - lava::attachment, 29
  - lava::block, 40
  - lava::buffer, 47
  - lava::command, 58
  - lava::compute\_pipeline, 62
  - lava::descriptor, 80
  - lava::descriptor::binding, 34
  - lava::descriptor::pool, 231
  - lava::device, 89
  - lava::image, 159
  - lava::layer, 186
  - lava::mesh\_template< T >, 198
  - lava::physical\_device, 215
  - lava::pipeline::shader\_stage, 287
  - lava::pipeline\_layout, 225
  - lava::render\_pass, 256
  - lava::render\_pipeline, 266
  - lava::render\_target, 277
  - lava::subpass, 296
  - lava::subpass\_dependency, 301
  - lava::texture, 318
- make\_primitive\_indices\_cube
  - mesh.hpp, 584
- make\_primitive\_normals\_cube
  - mesh.hpp, 584
- make\_primitive\_positions\_cube
  - mesh.hpp, 584
- make\_primitive\_uvs\_cube
  - mesh.hpp, 585
- math.hpp
  - ceil\_div, 625
  - default\_color, 626
  - hash256, 625
  - perspective\_matrix, 625
- maximized
  - lava::window, 338
- memory.hpp
  - create\_allocator, 441
  - find\_memory\_type, 441
  - find\_memory\_type\_with\_properties, 441
- mesh.hpp
  - create\_mesh, 583
  - create\_mesh\_data, 583
  - make\_primitive\_indices\_cube, 584
  - make\_primitive\_normals\_cube, 584
  - make\_primitive\_positions\_cube, 584
  - make\_primitive\_uvs\_cube, 585
- misc.hpp
  - append, 484
  - begin, 484
  - contains, 485
  - end, 485
  - exists, 486
  - remove, 486
  - remove\_chars, 486
  - remove\_chars\_copy, 487
  - remove\_chars\_if\_not, 487
  - remove\_chars\_if\_not\_copy, 487
  - remove\_nondigit, 488
  - remove\_nondigit\_copy, 488
  - remove\_punctuation\_marks, 488
  - reverse, 489
  - trim, 489
  - trim\_copy, 489
  - trim\_end, 490
  - trim\_end\_copy, 490
  - trim\_start, 490
  - trim\_start\_copy, 490
- mount
  - lava::file\_system, 123
- mount\_base
  - lava::file\_system, 123
- mount\_res
  - lava::file\_system, 124
- move
  - lava::mesh\_template\_data< T >, 199
  - lava::rect, 249
- moving
  - lava::camera, 55
- next
  - lava::ids, 154
- next\_pow\_2
  - data.hpp, 475
- now
  - frame.hpp, 533
- one\_time\_submit
  - device.hpp, 429
- one\_time\_submit\_pool
  - device.hpp, 429
- open
  - lava::file, 112
- opened
  - lava::file, 113
- operator bool
  - lava::vk\_result, 329
- operator<
  - lava::queue, 243
  - lava::telegram, 313
- operator()
  - lava::pair\_hash, 209
  - lava::props, 239
- operator==
  - lava::telegram, 314
  - lava::vertex, 328
- parse
  - lava::props, 240
- parse\_benchmark
  - benchmark.hpp, 349
- perspective\_matrix
  - math.hpp, 625
- physical\_device
  - lava::physical\_device, 212



- pipeline
  - lava::pipeline, [218](#)
- pipeline.hpp
  - create\_pipeline\_shader\_stage, [459](#)
- pressed
  - lava::gamepad, [135](#)
  - lava::key\_event, [183](#)
  - lava::mouse\_button\_event, [205](#)
- process
  - lava::block, [40](#)
  - lava::render\_pass, [256](#)
  - lava::subpass, [296](#)
- pseudorandom\_generator
  - lava::pseudorandom\_generator, [241](#)
- queue.hpp
  - add\_dedicated\_queues, [447](#)
  - add\_queues, [447](#)
  - default\_queue\_flags, [448](#)
  - set\_all\_queues, [447](#)
  - set\_default\_queues, [448](#)
  - verify\_queues, [448](#)
- queues
  - lava::device, [89](#)
- random
  - random.hpp, [628](#)
- random.hpp
  - random, [628](#)
- read
  - lava::file, [113](#)
- read\_file
  - file\_utils.hpp, [527](#)
- ready
  - lava::file\_system, [124](#)
  - lava::frame, [131](#)
  - lava::gamepad, [136](#)
  - lava::image\_data, [163](#)
  - lava::imgui, [168](#)
  - lava::pipeline, [220](#)
- realloc\_data
  - data.hpp, [475](#)
- rect
  - lava::rect, [248](#)
- released
  - lava::key\_event, [183](#)
  - lava::mouse\_button\_event, [206](#)
- reload
  - lava::mesh\_template< T >, [198](#)
- reload\_request
  - lava::render\_target, [277](#)
  - lava::swapchain, [310](#)
- reload\_shader
  - lava::producer, [235](#)
- remove
  - lava::frame, [131](#)
  - lava::gamepad\_manager, [137](#)
  - lava::id\_listeners< T >, [149](#)
  - lava::id\_registry< T, Meta >, [152](#)
  - lava::input, [171](#)
  - lava::json\_file, [181](#)
  - lava::layer\_list, [189](#)
  - lava::platform, [228](#)
  - lava::props, [240](#)
  - lava::render\_pass, [257](#)
  - lava::subpass, [296](#)
  - misc.hpp, [486](#)
- remove\_callback
  - lava::render\_target, [277](#)
  - lava::swapchain, [310](#)
- remove\_chars
  - misc.hpp, [486](#)
- remove\_chars\_copy
  - misc.hpp, [487](#)
- remove\_chars\_if\_not
  - misc.hpp, [487](#)
- remove\_chars\_if\_not\_copy
  - misc.hpp, [487](#)
- remove\_cmd
  - lava::block, [41](#)
- remove\_command
  - lava::block, [41](#)
- remove\_dispatch
  - lava::message\_dispatcher, [202](#)
- remove\_existing\_path
  - file\_utils.hpp, [527](#)
- remove\_id\_map
  - id.hpp, [479](#)
- remove\_nondigit
  - misc.hpp, [488](#)
- remove\_nondigit\_copy
  - misc.hpp, [488](#)
- remove\_punctuation\_marks
  - misc.hpp, [488](#)
- render\_pass
  - lava::render\_pass, [252](#)
- render\_pipeline
  - lava::render\_pipeline, [262](#)
- render\_pipeline.hpp
  - create\_pipeline\_color\_blend\_attachment, [466](#)
- render\_target.hpp
  - create\_target, [553](#)
  - create\_target\_no\_triple\_buffer, [553](#)
  - create\_target\_v\_sync, [554](#)
- repeated
  - lava::key\_event, [184](#)
- resizable
  - lava::window, [338](#)
- resize
  - lava::render\_target, [278](#)
  - lava::swapchain, [310](#)
- resize\_request
  - lava::window, [339](#)
- reverse
  - misc.hpp, [489](#)
- run
  - lava::driver, [103](#)

- lava::frame, 132
- save
  - lava::json\_file, 181
- save\_title
  - lava::window, 339
- scale
  - lava::hex\_cell, 141
  - lava::mesh\_template\_data< T >, 200
- scale\_vector
  - lava::mesh\_template\_data< T >, 200
- scoped\_label
  - lava::scoped\_label< T >, 284
- screenshot
  - lava::app, 23
- seek
  - lava::file, 113
- sem\_version\_string
  - log.hpp, 619
- semantic\_version\_string
  - log.hpp, 620
- send\_message
  - lava::message\_dispatcher, 202
  - lava::telegraph, 315
- set
  - lava::compute\_pipeline, 64
  - lava::data, 73
  - lava::descriptor::binding, 34
  - lava::global\_logger, 139
  - lava::json\_file, 181
  - lava::render\_pipeline, 267
  - lava::subpass, 297
  - lava::tooltip\_list, 324
- set\_acceleration\_structure\_name
  - debug\_utils.hpp, 393
- set\_acceleration\_structure\_nv\_name
  - debug\_utils.hpp, 394
- set\_acceleration\_structure\_nv\_tag
  - debug\_utils.hpp, 394
- set\_acceleration\_structure\_tag
  - debug\_utils.hpp, 394
- set\_access\_mask
  - lava::subpass\_dependency, 303
- set\_active
  - lava::block, 41
  - lava::camera, 55
  - lava::imgui, 168
  - lava::pipeline, 220
  - lava::subpass, 297
- set\_all\_queues
  - queue.hpp, 447
- set\_allocator
  - lava::device, 89
- set\_aspect\_mask
  - lava::image, 160
- set\_auto\_bind
  - lava::pipeline, 221
- set\_auto\_line\_width
  - lava::render\_pipeline, 267
- set\_auto\_size
  - lava::render\_pipeline, 267
- set\_buffer\_name
  - debug\_utils.hpp, 394
- set\_buffer\_tag
  - debug\_utils.hpp, 395
- set\_buffer\_view\_name
  - debug\_utils.hpp, 395
- set\_buffer\_view\_tag
  - debug\_utils.hpp, 395
- set\_callbacks
  - lava::memory, 191
- set\_clear\_color
  - lava::render\_pass, 257
- set\_clear\_values
  - lava::render\_pass, 257
- set\_color\_attachment
  - lava::subpass, 297
- set\_color\_attachments
  - lava::subpass, 298
- set\_command\_buffer\_name
  - debug\_utils.hpp, 395
- set\_command\_buffer\_tag
  - debug\_utils.hpp, 396
- set\_command\_pool\_name
  - debug\_utils.hpp, 396
- set\_command\_pool\_tag
  - debug\_utils.hpp, 396
- set\_component
  - lava::image, 160
- set\_count
  - lava::descriptor::binding, 34
- set\_data
  - lava::image\_data, 163
  - lava::mesh\_template< T >, 198
- set\_debug\_report\_callback\_name
  - debug\_utils.hpp, 396
- set\_debug\_report\_callback\_tag
  - debug\_utils.hpp, 397
- set\_debug\_utils\_messenger\_name
  - debug\_utils.hpp, 397
- set\_debug\_utils\_messenger\_tag
  - debug\_utils.hpp, 397
- set\_decorated
  - lava::window, 339
- set\_default\_queues
  - queue.hpp, 448
- set\_deferred\_operation\_name
  - debug\_utils.hpp, 397
- set\_deferred\_operation\_tag
  - debug\_utils.hpp, 398
- set\_dependency\_flags
  - lava::subpass\_dependency, 303
- set\_depth\_compare\_op
  - lava::render\_pipeline, 267
- set\_depth\_stencil\_attachment
  - lava::subpass, 298
- set\_depth\_test\_and\_write

- lava::render\_pipeline, 268
- set\_descriptor\_pool\_name
  - debug\_utils.hpp, 398
- set\_descriptor\_pool\_tag
  - debug\_utils.hpp, 398
- set\_descriptor\_set\_layout\_name
  - debug\_utils.hpp, 398
- set\_descriptor\_set\_layout\_tag
  - debug\_utils.hpp, 399
- set\_descriptor\_set\_name
  - debug\_utils.hpp, 399
- set\_descriptor\_set\_tag
  - debug\_utils.hpp, 399
- set\_descriptor\_update\_template\_name
  - debug\_utils.hpp, 399
- set\_descriptor\_update\_template\_tag
  - debug\_utils.hpp, 400
- set\_device\_memory\_name
  - debug\_utils.hpp, 400
- set\_device\_memory\_tag
  - debug\_utils.hpp, 400
- set\_device\_name
  - debug\_utils.hpp, 400
- set\_device\_tag
  - debug\_utils.hpp, 401
- set\_display\_mode\_name
  - debug\_utils.hpp, 401
- set\_display\_mode\_tag
  - debug\_utils.hpp, 401
- set\_display\_name
  - debug\_utils.hpp, 401
- set\_display\_tag
  - debug\_utils.hpp, 402
- set\_dst\_access\_mask
  - lava::subpass\_dependency, 303
- set\_dst\_stage\_mask
  - lava::subpass\_dependency, 303
- set\_dst\_subpass
  - lava::subpass\_dependency, 304
- set\_dynamic\_states
  - lava::render\_pipeline, 268
- set\_event\_name
  - debug\_utils.hpp, 402
- set\_event\_tag
  - debug\_utils.hpp, 402
- set\_fence\_name
  - debug\_utils.hpp, 402
- set\_fence\_tag
  - debug\_utils.hpp, 403
- set\_filename
  - lava::props, 240
- set\_final\_layout
  - lava::attachment, 29
- set\_flags
  - lava::image, 160
- set\_floating
  - lava::window, 339
- set\_format
  - lava::attachment, 29
- set\_framebuffer\_name
  - debug\_utils.hpp, 403
- set\_framebuffer\_tag
  - debug\_utils.hpp, 403
- set\_fullscreen
  - lava::window, 340
- set\_icon
  - lava::window, 340
- set\_image\_layout
  - format.hpp, 575
- set\_image\_name
  - debug\_utils.hpp, 403
- set\_image\_tag
  - debug\_utils.hpp, 404
- set\_image\_view\_name
  - debug\_utils.hpp, 404
- set\_image\_view\_tag
  - debug\_utils.hpp, 404
- set\_indirect\_commands\_layout\_name
  - debug\_utils.hpp, 404
- set\_indirect\_commands\_layout\_tag
  - debug\_utils.hpp, 405
- set\_ini\_file
  - lava::imgui, 169
- set\_initial\_layout
  - lava::attachment, 30
- set\_input\_attachment
  - lava::subpass, 298, 299
- set\_input\_attachments
  - lava::subpass, 299
- set\_input\_topology
  - lava::render\_pipeline, 268
- set\_instance\_name
  - debug\_utils.hpp, 405
- set\_instance\_tag
  - debug\_utils.hpp, 405
- set\_json
  - lava::app\_config, 27
  - lava::configurable, 66
  - lava::props, 240
- set\_layer\_count
  - lava::image, 160
- set\_layout
  - lava::image, 161
  - lava::pipeline, 221
- set\_layouts
  - lava::attachment, 30
- set\_level\_count
  - lava::image, 161
- set\_line\_width
  - lava::render\_pipeline, 268, 269
- set\_load\_op
  - lava::attachment, 30
- set\_mouse\_position
  - lava::input, 171
  - lava::window, 340
- set\_movement\_keys

- lava::camera, 55
- set\_name
  - debug\_utils.hpp, 405
- set\_object\_name
  - debug\_utils.hpp, 406
- set\_object\_tag
  - debug\_utils.hpp, 406
- set\_op
  - lava::attachment, 30
- set\_performance\_configuration\_name
  - debug\_utils.hpp, 406
- set\_performance\_configuration\_tag
  - debug\_utils.hpp, 407
- set\_physical\_device\_name
  - debug\_utils.hpp, 407
- set\_physical\_device\_tag
  - debug\_utils.hpp, 407
- set\_pipeline\_cache\_name
  - debug\_utils.hpp, 407
- set\_pipeline\_cache\_tag
  - debug\_utils.hpp, 408
- set\_pipeline\_layout\_name
  - debug\_utils.hpp, 408
- set\_pipeline\_layout\_tag
  - debug\_utils.hpp, 408
- set\_pipeline\_name
  - debug\_utils.hpp, 408
- set\_pipeline\_tag
  - debug\_utils.hpp, 409
- set\_position
  - lava::window, 340
- set\_preserve\_attachments
  - lava::subpass, 299
- set\_private\_data\_slot\_name
  - debug\_utils.hpp, 409
- set\_private\_data\_slot\_tag
  - debug\_utils.hpp, 409
- set\_query\_pool\_name
  - debug\_utils.hpp, 409
- set\_query\_pool\_tag
  - debug\_utils.hpp, 410
- set\_queue\_name
  - debug\_utils.hpp, 410
- set\_queue\_tag
  - debug\_utils.hpp, 410
- set\_rasterization\_cull\_mode
  - lava::render\_pipeline, 269
- set\_rasterization\_front\_face
  - lava::render\_pipeline, 269
- set\_rasterization\_polygon\_mode
  - lava::render\_pipeline, 269
- set\_render\_pass
  - lava::render\_pipeline, 270
- set\_render\_pass\_name
  - debug\_utils.hpp, 410
- set\_render\_pass\_tag
  - debug\_utils.hpp, 411
- set\_resizable
  - lava::window, 341
- set\_resolve\_attachment
  - lava::subpass, 299, 300
- set\_resolve\_attachments
  - lava::subpass, 300
- set\_sampler\_name
  - debug\_utils.hpp, 411
- set\_sampler\_tag
  - debug\_utils.hpp, 411
- set\_sampler\_ycbcr\_conversion\_name
  - debug\_utils.hpp, 411
- set\_sampler\_ycbcr\_conversion\_tag
  - debug\_utils.hpp, 412
- set\_samplers
  - lava::descriptor::binding, 35
- set\_samples
  - lava::attachment, 31
- set\_save\_name
  - lava::window, 341
- set\_scissor
  - lava::render\_pipeline, 270
- set\_seed
  - lava::pseudorandom\_generator, 242
- set\_semaphore\_name
  - debug\_utils.hpp, 412
- set\_semaphore\_tag
  - debug\_utils.hpp, 412
- set\_shader\_module\_name
  - debug\_utils.hpp, 412
- set\_shader\_module\_tag
  - debug\_utils.hpp, 413
- set\_shader\_stage
  - lava::compute\_pipeline, 64
- set\_size
  - lava::rect, 250
  - lava::window, 341
- set\_sizing
  - lava::render\_pipeline, 270
- set\_src\_access\_mask
  - lava::subpass\_dependency, 304
- set\_src\_stage\_mask
  - lava::subpass\_dependency, 304
- set\_src\_subpass
  - lava::subpass\_dependency, 304
- set\_stage
  - lava::pipeline::shader\_stage, 289
- set\_stage\_flags
  - lava::descriptor::binding, 35
- set\_stage\_mask
  - lava::subpass\_dependency, 305
- set\_state
  - lava::window, 341
- set\_stencil\_load\_op
  - lava::attachment, 31
- set\_stencil\_op
  - lava::attachment, 31
- set\_stencil\_store\_op
  - lava::attachment, 31

- set\_store\_op
  - lava::attachment, 32
- set\_subpass
  - lava::render\_pipeline, 270
  - lava::subpass\_dependency, 305
- set\_surface\_name
  - debug\_utils.hpp, 413
- set\_surface\_tag
  - debug\_utils.hpp, 413
- set\_swapchain\_name
  - debug\_utils.hpp, 413
- set\_swapchain\_tag
  - debug\_utils.hpp, 414
- set\_tag
  - debug\_utils.hpp, 414
- set\_tiling
  - lava::image, 161
- set\_title
  - lava::window, 342
- set\_type
  - lava::descriptor::binding, 35
- set\_usage
  - lava::image, 161
- set\_use\_custom\_cpu\_callbacks
  - lava::memory, 191
- set\_validation\_cache\_name
  - debug\_utils.hpp, 414
- set\_validation\_cache\_tag
  - debug\_utils.hpp, 415
- set\_vertex\_input\_attribute
  - lava::render\_pipeline, 271
- set\_vertex\_input\_attributes
  - lava::render\_pipeline, 271
- set\_vertex\_input\_binding
  - lava::render\_pipeline, 271
- set\_vertex\_input\_bindings
  - lava::render\_pipeline, 271
- set\_view\_type
  - lava::image, 162
- set\_viewport
  - lava::render\_pipeline, 272
- set\_viewport\_and\_scissor
  - lava::render\_pipeline, 272
- set\_wait\_for\_events
  - lava::frame, 132
- set\_window\_icon
  - config.hpp, 354
- setup
  - lava::app, 23
  - lava::engine, 108
  - lava::imgui, 169
  - lava::message\_dispatcher, 203
  - lava::pipeline, 221
  - lava::thread\_pool, 321
  - log.hpp, 620
- setup\_imgui\_font
  - imgui.hpp, 372
- setup\_imgui\_font\_icons
  - imgui.hpp, 372
- show\_save\_title
  - lava::window, 342
- shut\_down
  - lava::frame, 132
- singleton
  - lava::gamepad\_manager, 138
  - lava::global\_logger, 139
  - lava::instance, 176
- size
  - lava::image\_data, 163
- sleep
  - thread.hpp, 633, 634
- stage
  - lava::stage, 290
  - lava::staging, 291
  - lava::texture, 319
- STR
  - driver.hpp, 538
- str
  - types.hpp, 506
- STR\_
  - driver.hpp, 538
- subtract
  - lava::hex\_cell, 141
- support\_blit
  - format.hpp, 576
- support\_vertex\_buffer\_format
  - format.hpp, 576
- supported
  - lava::physical\_device, 215
- surface\_supported
  - lava::device, 90
  - lava::physical\_device, 215
  - lava::swapchain, 311
- swapchain\_supported
  - lava::physical\_device, 215
- switch\_config
  - lava::app, 24
- switch\_mode
  - lava::window, 342
- switch\_mode\_request
  - lava::window, 342
- teardown
  - log.hpp, 620
- telegram
  - lava::telegram, 313
- tell
  - lava::file, 114
- thread.hpp
  - sleep, 633, 634
- time.hpp
  - get\_current\_time, 494
  - get\_current\_timestamp, 494
  - get\_current\_timestamp\_ms, 494
  - get\_current\_timestamp\_us, 495
  - timestamp, 495
  - to\_delta, 495

- to\_dt, [496](#)
- to\_ms, [496](#)
- to\_sec, [496](#)
- to\_sec\_fix, [497](#)
- timestamp
  - time.hpp, [495](#)
- to\_api\_version
  - base.hpp, [386](#)
- to\_char
  - types.hpp, [507](#)
- to\_delta
  - time.hpp, [495](#)
- to\_dt
  - time.hpp, [496](#)
- to\_i32
  - types.hpp, [507](#)
- to\_i64
  - types.hpp, [507](#)
- to\_id
  - id.hpp, [480](#)
- to\_index
  - types.hpp, [508](#)
- to\_ms
  - time.hpp, [496](#)
- to\_pair
  - lava::hex\_cell, [141](#)
- to\_pixel
  - lava::hex\_grid, [143](#)
- to\_r32
  - types.hpp, [508](#)
- to\_r64
  - types.hpp, [508](#)
- to\_sec
  - time.hpp, [496](#)
- to\_sec\_fix
  - time.hpp, [497](#)
- to\_size\_t
  - types.hpp, [509](#)
- to\_string
  - base.hpp, [386](#)
  - hex.hpp, [609](#)
  - input.hpp, [546](#)
  - lava::id, [148](#)
  - log.hpp, [620](#), [621](#)
- to\_ui32
  - types.hpp, [509](#)
- to\_ui64
  - types.hpp, [509](#)
- to\_version
  - base.hpp, [386](#)
  - version.hpp, [514](#)
- to\_vk\_version
  - base.hpp, [387](#)
- tooltip
  - lava::tooltip, [323](#)
- transfer\_queue
  - lava::device, [90](#)
- transfer\_queues
  - lava::device, [90](#)
- trim
  - misc.hpp, [489](#)
- trim\_copy
  - misc.hpp, [489](#)
- trim\_end
  - misc.hpp, [490](#)
- trim\_end\_copy
  - misc.hpp, [490](#)
- trim\_start
  - misc.hpp, [490](#)
- trim\_start\_copy
  - misc.hpp, [490](#)
- triple\_buffer
  - lava::app, [25](#)
  - lava::swapchain, [311](#)
- types.hpp
  - c16, [503](#)
  - c32, [503](#)
  - c8, [503](#)
  - ENUM\_FLAG\_OPERATORS, [502](#)
  - hash\_combine, [505](#)
  - hash\_value, [506](#)
  - i16, [503](#)
  - i32, [503](#)
  - i64, [504](#)
  - i8, [504](#)
  - str, [506](#)
  - to\_char, [507](#)
  - to\_i32, [507](#)
  - to\_i64, [507](#)
  - to\_index, [508](#)
  - to\_r32, [508](#)
  - to\_r64, [508](#)
  - to\_size\_t, [509](#)
  - to\_ui32, [509](#)
  - to\_ui64, [509](#)
  - uc16, [504](#)
  - uc32, [504](#)
  - uc8, [504](#)
  - ui16, [504](#)
  - ui32, [505](#)
  - ui64, [505](#)
  - ui8, [505](#)
- u\_data
  - lava::u\_data, [326](#)
- uc16
  - types.hpp, [504](#)
- uc32
  - types.hpp, [504](#)
- uc8
  - types.hpp, [504](#)
- ui16
  - types.hpp, [504](#)
- ui32
  - types.hpp, [505](#)
- ui64
  - types.hpp, [505](#)

- ui8
  - types.hpp, 505
- unload
  - lava::props, 241
- update
  - lava::gamepad, 136
  - lava::hex\_grid, 143
  - lava::id\_registry< T, Meta >, 153
  - lava::message\_dispatcher, 203
- update\_view
  - lava::camera, 55, 56
- upload
  - lava::texture, 319
- upload\_fonts
  - lava::imgui, 169
- v\_sync
  - lava::app, 25
  - lava::swapchain, 311
- valid
  - lava::allocator, 17
  - lava::buffer, 47
  - lava::camera, 56
  - lava::id, 148
  - lava::queue, 243
- value
  - lava::gamepad, 136
- verify\_queues
  - lava::device::create\_param, 69
  - queue.hpp, 448
- version.hpp
  - to\_version, 514
- version\_string
  - log.hpp, 622
- visible
  - lava::window, 342
- vk\_version\_to\_string
  - base.hpp, 387
- vkAcquireNextImageKHR
  - lava::device\_table, 92
- vkAllocateCommandBuffers
  - lava::device\_table, 92, 93
- vkCreateCommandPool
  - lava::device\_table, 93
- vkCreateFence
  - lava::device\_table, 94
- vkCreateImageView
  - lava::device\_table, 94
- vkCreateSampler
  - lava::device\_table, 95
- vkCreateSemaphore
  - lava::device\_table, 95
- vkCreateShaderModule
  - lava::device\_table, 96
- vkCreateSwapchainKHR
  - lava::device\_table, 96
- vkDestroyCommandPool
  - lava::device\_table, 97
- vkDestroyFence
  - lava::device\_table, 97
- vkDestroyImageView
  - lava::device\_table, 97
- vkDestroySampler
  - lava::device\_table, 97
- vkDestroySemaphore
  - lava::device\_table, 98
- vkDestroySwapchainKHR
  - lava::device\_table, 98
- vkFreeCommandBuffers
  - lava::device\_table, 98
- vkGetSwapchainImagesKHR
  - lava::device\_table, 98
- vkQueuePresentKHR
  - lava::device\_table, 99
- vkQueueSubmit
  - lava::device\_table, 99
- vkResetFences
  - lava::device\_table, 99
- vkUpdateDescriptorSets
  - lava::device\_table, 99–101
- vkWaitForFences
  - lava::device\_table, 101
- wait\_for\_idle
  - lava::device, 90
- waiting\_for\_events
  - lava::frame, 132
- window
  - lava::window, 333
- window.hpp
  - create\_surface, 560
  - get\_window, 560
- writable
  - lava::file, 114
- write
  - lava::file, 114
- write\_file
  - file\_utils.hpp, 527
- write\_frames\_json
  - benchmark.hpp, 349
- write\_image.hpp
  - write\_image\_png, 381
- write\_image\_png
  - write\_image.hpp, 381