

# The Dungeon Throne

1.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>The Dungeon Throne</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>7</b>
3.1	Class Hierarchy . . . . .	7
<b>4</b>	<b>Class Index</b>	<b>11</b>
4.1	Class List . . . . .	11
<b>5</b>	<b>Namespace Documentation</b>	<b>17</b>
5.1	action Namespace Reference . . . . .	17
5.1.1	Detailed Description . . . . .	17
5.1.2	Function Documentation . . . . .	18
5.1.2.1	BUILD_TAB() . . . . .	18
5.1.2.2	CAST_SPELL_1() . . . . .	18
5.1.2.3	CAST_SPELL_2() . . . . .	18
5.1.2.4	CAST_SPELL_3() . . . . .	18
5.1.2.5	CAST_SPELL_4() . . . . .	18
5.1.2.6	MENU_TAB() . . . . .	18
5.1.2.7	NEXT() . . . . .	18
5.1.2.8	PREV() . . . . .	19
5.1.2.9	QUICK_LOAD() . . . . .	19
5.1.2.10	QUICK_SAVE() . . . . .	19

5.1.2.11	RESET_CAMERA()	19
5.1.2.12	SPELL_TAB()	19
5.2	AIHelper Namespace Reference	19
5.2.1	Detailed Description	19
5.2.2	Function Documentation	19
5.2.2.1	get_blueprint(EntitySystem &, tdt::uint)	19
5.2.2.2	get_state(EntitySystem &, tdt::uint)	20
5.2.2.3	set_blueprint(EntitySystem &, tdt::uint, const std::string &)	20
5.2.2.4	set_state(EntitySystem &, tdt::uint, ENTITY_STATE::VAL)	20
5.3	CombatHelper Namespace Reference	20
5.3.1	Detailed Description	21
5.3.2	Function Documentation	21
5.3.2.1	get_atk_type(EntitySystem &, tdt::uint)	21
5.3.2.2	get_cooldown(EntitySystem &, tdt::uint)	22
5.3.2.3	get_dmg(tdt::uint, tdt::uint)	22
5.3.2.4	get_dmg_range(EntitySystem &, tdt::uint)	22
5.3.2.5	get_projectile_blueprint(EntitySystem &, tdt::uint)	22
5.3.2.6	get_range(EntitySystem &, tdt::uint)	23
5.3.2.7	get_target(EntitySystem &, tdt::uint)	23
5.3.2.8	in_range(EntitySystem &, tdt::uint, tdt::uint)	23
5.3.2.9	set_atk_type(EntitySystem &, tdt::uint, ATTACK_TYPE)	23
5.3.2.10	set_cooldown(EntitySystem &, tdt::uint, tdt::real)	24
5.3.2.11	set_dmg_range(EntitySystem &, tdt::uint, tdt::uint, tdt::uint)	24
5.3.2.12	set_projectile_blueprint(EntitySystem &, tdt::uint, const std::string &)	24
5.3.2.13	set_range(EntitySystem &, tdt::uint, tdt::real)	24
5.3.2.14	set_target(EntitySystem &, tdt::uint, tdt::uint)	25
5.4	CommandHelper Namespace Reference	25
5.4.1	Detailed Description	25
5.4.2	Function Documentation	25
5.4.2.1	command_to_attack(EntitySystem &, SelectionBox &)	25

5.4.2.2	<a href="#">command_to_fall_back(EntitySystem &amp;)</a>	26
5.4.2.3	<a href="#">command_to_mine(EntitySystem &amp;, SelectionBox &amp;)</a>	26
5.4.2.4	<a href="#">command_to_reposition(EntitySystem &amp;, Ogre::Real, Ogre::Real)</a>	26
5.4.2.5	<a href="#">command_to_return_gold(EntitySystem &amp;, CombatSystem &amp;)</a>	26
5.4.2.6	<a href="#">set_command(EntitySystem &amp;, tdt::uint, COMMAND_TYPE, bool=true)</a>	27
5.4.2.7	<a href="#">test_command(EntitySystem &amp;, tdt::uint, COMMAND_TYPE)</a>	27
5.5	<a href="#">ConstructorHelper Namespace Reference</a>	27
5.5.1	<a href="#">Detailed Description</a>	28
5.5.2	<a href="#">Function Documentation</a>	28
5.5.2.1	<a href="#">call(EntitySystem &amp;, tdt::uint)</a>	28
5.5.2.2	<a href="#">get_blueprint(EntitySystem &amp;, tdt::uint)</a>	28
5.5.2.3	<a href="#">set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</a>	28
5.6	<a href="#">CounterHelper Namespace Reference</a>	28
5.6.1	<a href="#">Detailed Description</a>	29
5.6.2	<a href="#">Function Documentation</a>	29
5.6.2.1	<a href="#">decrement(EntitySystem &amp;, tdt::uint)</a>	29
5.6.2.2	<a href="#">get_curr_value(EntitySystem &amp;, tdt::uint)</a>	29
5.6.2.3	<a href="#">get_max_value(EntitySystem &amp;, tdt::uint)</a>	29
5.6.2.4	<a href="#">increment(EntitySystem &amp;, tdt::uint)</a>	30
5.6.2.5	<a href="#">set_curr_value(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	30
5.6.2.6	<a href="#">set_max_value(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	30
5.7	<a href="#">DestructorHelper Namespace Reference</a>	30
5.7.1	<a href="#">Detailed Description</a>	31
5.7.2	<a href="#">Function Documentation</a>	31
5.7.2.1	<a href="#">destroy(EntitySystem &amp;, tdt::uint, bool=false, tdt::uint=Component::NO_ENTITY)</a>	31
5.7.2.2	<a href="#">get_blueprint(EntitySystem &amp;, tdt::uint)</a>	31
5.7.2.3	<a href="#">set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</a>	31
5.8	<a href="#">EventHandlerHelper Namespace Reference</a>	32
5.8.1	<a href="#">Detailed Description</a>	32
5.8.2	<a href="#">Function Documentation</a>	32

5.8.2.1	<code>add_possible_event(EntitySystem &amp;, tdt::uint, EVENT_TYPE)</code>	32
5.8.2.2	<code>can_handle(EntitySystem &amp;, tdt::uint, EVENT_TYPE)</code>	33
5.8.2.3	<code>delete_possible_event(EntitySystem &amp;, tdt::uint, EVENT_TYPE)</code>	33
5.8.2.4	<code>get_handler(EntitySystem &amp;, tdt::uint)</code>	33
5.8.2.5	<code>set_handler(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	33
5.9	EventHelper Namespace Reference	34
5.9.1	Detailed Description	34
5.9.2	Function Documentation	34
5.9.2.1	<code>get_event_handler(EntitySystem &amp;, tdt::uint)</code>	34
5.9.2.2	<code>get_event_type(EntitySystem &amp;, tdt::uint)</code>	35
5.9.2.3	<code>get_radius(EntitySystem &amp;, tdt::uint)</code>	35
5.9.2.4	<code>get_target(EntitySystem &amp;, tdt::uint)</code>	35
5.9.2.5	<code>is_active(EntitySystem &amp;, tdt::uint)</code>	35
5.9.2.6	<code>set_active(EntitySystem &amp;, tdt::uint, bool=true)</code>	36
5.9.2.7	<code>set_event_handler(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	36
5.9.2.8	<code>set_event_type(EntitySystem &amp;, tdt::uint, EVENT_TYPE)</code>	36
5.9.2.9	<code>set_radius(EntitySystem &amp;, tdt::uint, tdt::real)</code>	36
5.9.2.10	<code>set_target(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	37
5.10	ExperienceValueHelper Namespace Reference	37
5.10.1	Detailed Description	37
5.10.2	Function Documentation	37
5.10.2.1	<code>decrease(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	37
5.10.2.2	<code>get(EntitySystem &amp;, tdt::uint)</code>	38
5.10.2.3	<code>increase(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	38
5.10.2.4	<code>set(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	38
5.11	ExplosionHelper Namespace Reference	38
5.11.1	Detailed Description	39
5.11.2	Function Documentation	39
5.11.2.1	<code>get_curr_radius(EntitySystem &amp;, tdt::uint)</code>	39
5.11.2.2	<code>get_delta(EntitySystem &amp;, tdt::uint)</code>	39

5.11.2.3	<a href="#">get_max_radius(EntitySystem &amp;, tdt::uint)</a>	39
5.11.2.4	<a href="#">increase_curr_radius(EntitySystem &amp;, tdt::uint, tdt::real)</a>	40
5.11.2.5	<a href="#">set_delta(EntitySystem &amp;, tdt::uint, tdt::real)</a>	40
5.11.2.6	<a href="#">set_max_radius(EntitySystem &amp;, tdt::uint, tdt::real)</a>	40
5.12	<a href="#">FactionHelper Namespace Reference</a>	41
5.12.1	<a href="#">Detailed Description</a>	41
5.12.2	<a href="#">Function Documentation</a>	41
5.12.2.1	<a href="#">get_faction(EntitySystem &amp;, tdt::uint)</a>	41
5.12.2.2	<a href="#">get_faction_name(EntitySystem &amp;, tdt::uint)</a>	41
5.12.2.3	<a href="#">set_faction(EntitySystem &amp;, tdt::uint, FACTION)</a>	41
5.13	<a href="#">GoldHelper Namespace Reference</a>	42
5.13.1	<a href="#">Detailed Description</a>	42
5.13.2	<a href="#">Function Documentation</a>	42
5.13.2.1	<a href="#">add_gold(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	42
5.13.2.2	<a href="#">get_curr_gold(EntitySystem &amp;, tdt::uint)</a>	43
5.13.2.3	<a href="#">get_max_gold(EntitySystem &amp;, tdt::uint)</a>	43
5.13.2.4	<a href="#">gold_full(EntitySystem &amp;, tdt::uint)</a>	43
5.13.2.5	<a href="#">set_curr_gold(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	43
5.13.2.6	<a href="#">set_max_gold(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	44
5.13.2.7	<a href="#">sub_gold(EntitySystem &amp;, tdt::uint, tdt::uint, bool=true)</a>	44
5.13.2.8	<a href="#">transfer_all_gold(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	44
5.14	<a href="#">GraphicsHelper Namespace Reference</a>	45
5.14.1	<a href="#">Detailed Description</a>	46
5.14.2	<a href="#">Enumeration Type Documentation</a>	46
5.14.2.1	<a href="#">PLANE</a>	46
5.14.3	<a href="#">Function Documentation</a>	46
5.14.3.1	<a href="#">apply_scale(EntitySystem &amp;, tdt::uint)</a>	46
5.14.3.2	<a href="#">collide(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	46
5.14.3.3	<a href="#">get_bounds(EntitySystem &amp;, tdt::uint)</a>	46
5.14.3.4	<a href="#">get_manual_scaling(EntitySystem &amp;, tdt::uint)</a>	47

5.14.3.5	<code>get_material(EntitySystem &amp;, tdt::uint)</code>	47
5.14.3.6	<code>get_mesh(EntitySystem &amp;, tdt::uint)</code>	47
5.14.3.7	<code>get_query_flags(EntitySystem &amp;, tdt::uint)</code>	47
5.14.3.8	<code>get_scale(EntitySystem &amp;, tdt::uint)</code>	48
5.14.3.9	<code>init_graphics_component(EntitySystem &amp;, Ogre::SceneManager &amp;, tdt::uint)</code>	48
5.14.3.10	<code>is_visible(EntitySystem &amp;, tdt::uint)</code>	48
5.14.3.11	<code>look_at(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	48
5.14.3.12	<code>rotate(EntitySystem &amp;, tdt::uint, tdt::real, PLANE=PLANE::Y)</code>	49
5.14.3.13	<code>set_manual_scaling(EntitySystem &amp;, tdt::uint, bool)</code>	49
5.14.3.14	<code>set_material(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	49
5.14.3.15	<code>set_mesh(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	50
5.14.3.16	<code>set_query_flags(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	50
5.14.3.17	<code>set_scale(EntitySystem &amp;, tdt::uint, const Ogre::Vector3 &amp;)</code>	50
5.14.3.18	<code>set_visible(EntitySystem &amp;, tdt::uint, bool)</code>	51
5.15	GridNodeHelper Namespace Reference	51
5.15.1	Detailed Description	51
5.15.2	Function Documentation	51
5.15.2.1	<code>area_free(EntitySystem &amp;, tdt::uint, tdt::uint=1)</code>	51
5.15.2.2	<code>get_board_coords(EntitySystem &amp;, tdt::uint)</code>	52
5.15.2.3	<code>get_manhattan_distance(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	52
5.15.2.4	<code>get_neighbours(EntitySystem &amp;, tdt::uint)</code>	52
5.15.2.5	<code>get_node_in_dir(EntitySystem &amp;, tdt::uint, int)</code>	53
5.15.2.6	<code>get_resident(EntitySystem &amp;, tdt::uint)</code>	53
5.15.2.7	<code>is_free(EntitySystem &amp;, tdt::uint)</code>	53
5.15.2.8	<code>set_free(EntitySystem &amp;, tdt::uint, bool)</code>	53
5.15.2.9	<code>set_free_selected(EntitySystem &amp;, SelectionBox &amp;, bool)</code>	54
5.15.2.10	<code>set_portal_neighbour(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	54
5.15.2.11	<code>set_resident(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	54
5.16	HealthHelper Namespace Reference	54
5.16.1	Detailed Description	55



5.16.2	Function Documentation	55
5.16.2.1	add_defense(EntitySystem &, tdt::uint, tdt::uint)	55
5.16.2.2	add_health(EntitySystem &, tdt::uint, tdt::uint)	56
5.16.2.3	buff(EntitySystem &, tdt::uint, tdt::uint)	56
5.16.2.4	debuff(EntitySystem &, tdt::uint, tdt::uint)	56
5.16.2.5	get_defense(EntitySystem &, tdt::uint)	56
5.16.2.6	get_health(EntitySystem &, tdt::uint)	57
5.16.2.7	get_regen(EntitySystem &, tdt::uint)	57
5.16.2.8	heal(EntitySystem &, tdt::uint)	57
5.16.2.9	is_alive(EntitySystem &, tdt::uint)	57
5.16.2.10	set_alive(EntitySystem &, tdt::uint, bool)	58
5.16.2.11	set_defense(EntitySystem &, tdt::uint, tdt::uint)	58
5.16.2.12	set_health(EntitySystem &, tdt::uint, tdt::uint)	58
5.16.2.13	set_regen(EntitySystem &, tdt::uint, tdt::uint)	58
5.16.2.14	sub_defense(EntitySystem &, tdt::uint, tdt::uint)	59
5.16.2.15	sub_health(EntitySystem &, tdt::uint, tdt::uint, bool=false)	59
5.16.2.16	ubercharge(EntitySystem &, tdt::uint)	59
5.17	HomingHelper Namespace Reference	60
5.17.1	Detailed Description	60
5.17.2	Function Documentation	60
5.17.2.1	get_dmg(EntitySystem &, tdt::uint)	60
5.17.2.2	get_source(EntitySystem &, tdt::uint)	60
5.17.2.3	get_target(EntitySystem &, tdt::uint)	61
5.17.2.4	set_dmg(EntitySystem &, tdt::uint, tdt::uint)	61
5.17.2.5	set_source(EntitySystem &, tdt::uint, tdt::uint)	61
5.17.2.6	set_target(EntitySystem &, tdt::uint, tdt::uint)	61
5.18	InputHelper Namespace Reference	62
5.18.1	Detailed Description	62
5.18.2	Function Documentation	62
5.18.2.1	get_input_handler(EntitySystem &, tdt::uint)	62

5.18.2.2	<code>set_input_handler(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	62
5.19	level_generators Namespace Reference	63
5.19.1	Detailed Description	63
5.19.2	Typedef Documentation	63
5.19.2.1	DEFAULT_LEVEL_GENERATOR	63
5.20	LightHelper Namespace Reference	63
5.20.1	Detailed Description	64
5.20.2	Function Documentation	64
5.20.2.1	<code>init(EntitySystem &amp;, tdt::uint)</code>	64
5.20.2.2	<code>is_visible(EntitySystem &amp;, tdt::uint)</code>	64
5.20.2.3	<code>set_visible(EntitySystem &amp;, tdt::uint, bool)</code>	64
5.20.2.4	<code>toggle_visible(EntitySystem &amp;, tdt::uint)</code>	65
5.21	LimitedLifeSpanHelper Namespace Reference	65
5.21.1	Detailed Description	65
5.21.2	Function Documentation	65
5.21.2.1	<code>advance_curr_time(EntitySystem &amp;, tdt::uint, tdt::real)</code>	65
5.21.2.2	<code>get_curr_time(EntitySystem &amp;, tdt::uint)</code>	66
5.21.2.3	<code>get_max_time(EntitySystem &amp;, tdt::uint)</code>	66
5.21.2.4	<code>set_max_time(EntitySystem &amp;, tdt::uint, tdt::real)</code>	66
5.22	ManaCrystalHelper Namespace Reference	66
5.22.1	Detailed Description	67
5.22.2	Function Documentation	67
5.22.2.1	<code>get_capacity(EntitySystem &amp;, tdt::uint)</code>	67
5.22.2.2	<code>get_regen(EntitySystem &amp;, tdt::uint)</code>	67
5.22.2.3	<code>set_capacity(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	67
5.22.2.4	<code>set_regen(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	68
5.23	ManaHelper Namespace Reference	68
5.23.1	Detailed Description	68
5.23.2	Function Documentation	69
5.23.2.1	<code>add_mana(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	69

5.23.2.2	<code>get_mana(EntitySystem &amp;, tdt::uint)</code>	69
5.23.2.3	<code>get_max_mana(EntitySystem &amp;, tdt::uint)</code>	69
5.23.2.4	<code>get_regen(EntitySystem &amp;, tdt::uint)</code>	69
5.23.2.5	<code>set_mana(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	70
5.23.2.6	<code>set_max_mana(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	70
5.23.2.7	<code>set_regen(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	70
5.23.2.8	<code>sub_mana(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	70
5.24	MovementHelper Namespace Reference	71
5.24.1	Detailed Description	71
5.24.2	Function Documentation	71
5.24.2.1	<code>dir_to(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	71
5.24.2.2	<code>get_dir(EntitySystem &amp;, tdt::uint)</code>	72
5.24.2.3	<code>get_dir_back(EntitySystem &amp;, tdt::uint)</code>	72
5.24.2.4	<code>get_dir_left(EntitySystem &amp;, tdt::uint)</code>	72
5.24.2.5	<code>get_dir_right(EntitySystem &amp;, tdt::uint)</code>	72
5.24.2.6	<code>get_original_speed(EntitySystem &amp;, tdt::uint)</code>	73
5.24.2.7	<code>get_speed_modifier(EntitySystem &amp;, tdt::uint)</code>	73
5.24.2.8	<code>reset_speed(EntitySystem &amp;, tdt::uint)</code>	73
5.24.2.9	<code>set_original_speed(EntitySystem &amp;, tdt::uint, tdt::real)</code>	73
5.24.2.10	<code>set_speed_modifier(EntitySystem &amp;, tdt::uint, tdt::real)</code>	74
5.25	NameHelper Namespace Reference	74
5.25.1	Detailed Description	74
5.25.2	Function Documentation	74
5.25.2.1	<code>get_name(EntitySystem &amp;, tdt::uint)</code>	74
5.25.2.2	<code>set_name(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	75
5.26	NotificationHelper Namespace Reference	75
5.26.1	Detailed Description	75
5.26.2	Function Documentation	75
5.26.2.1	<code>advance_curr_time(EntitySystem &amp;, tdt::uint, tdt::real)</code>	75
5.26.2.2	<code>get_cooldown(EntitySystem &amp;, tdt::uint)</code>	76

5.26.2.3	<code>get_curr_time(EntitySystem &amp;, tdt::uint)</code>	76
5.26.2.4	<code>notify(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	76
5.26.2.5	<code>reset(EntitySystem &amp;, tdt::uint)</code>	76
5.26.2.6	<code>set_cooldown(EntitySystem &amp;, tdt::uint, tdt::real)</code>	77
5.27	OnHitHelper Namespace Reference	77
5.27.1	Detailed Description	77
5.27.2	Function Documentation	77
5.27.2.1	<code>call(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	77
5.27.2.2	<code>get_blueprint(EntitySystem &amp;, tdt::uint)</code>	78
5.27.2.3	<code>get_cooldown(EntitySystem &amp;, tdt::uint)</code>	78
5.27.2.4	<code>set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	78
5.27.2.5	<code>set_cooldown(EntitySystem &amp;, tdt::uint, tdt::real)</code>	78
5.28	PathfindingHelper Namespace Reference	79
5.28.1	Detailed Description	79
5.28.2	Function Documentation	79
5.28.2.1	<code>can_break(tdt::uint, const PathfindingComponent &amp;, tdt::uint)</code>	79
5.28.2.2	<code>get_cost(tdt::uint, const PathfindingComponent &amp;, tdt::uint, DIRECTION::VAL)</code>	79
5.28.2.3	<code>get_path(EntitySystem &amp;, tdt::uint)</code>	80
5.28.2.4	<code>get_pathfinding_blueprint(EntitySystem &amp;, tdt::uint)</code>	80
5.28.2.5	<code>set_pathfinding_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	80
5.29	PhysicsHelper Namespace Reference	81
5.29.1	Detailed Description	81
5.29.2	Function Documentation	81
5.29.2.1	<code>get_2d_position(EntitySystem &amp;, tdt::uint)</code>	81
5.29.2.2	<code>get_angle(Ogre::Vector3, Ogre::Vector3)</code>	82
5.29.2.3	<code>get_distance(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	82
5.29.2.4	<code>get_half_height(EntitySystem &amp;, tdt::uint)</code>	82
5.29.2.5	<code>get_position(EntitySystem &amp;, tdt::uint)</code>	83
5.29.2.6	<code>is_solid(EntitySystem &amp;, tdt::uint)</code>	83
5.29.2.7	<code>move_to(EntitySystem &amp;, tdt::uint, Ogre::Vector3)</code>	83

5.29.2.8	<a href="#">set_2d_position(EntitySystem &amp;, tdt::uint, Ogre::Vector2)</a>	83
5.29.2.9	<a href="#">set_half_height(EntitySystem &amp;, tdt::uint, tdt::real)</a>	84
5.29.2.10	<a href="#">set_position(EntitySystem &amp;, tdt::uint, const Ogre::Vector3 &amp;)</a>	84
5.29.2.11	<a href="#">set_solid(EntitySystem &amp;, tdt::uint, bool)</a>	84
5.30	<a href="#">PriceHelper Namespace Reference</a>	85
5.30.1	<a href="#">Detailed Description</a>	85
5.30.2	<a href="#">Function Documentation</a>	85
5.30.2.1	<a href="#">get_price(EntitySystem &amp;, tdt::uint)</a>	85
5.30.2.2	<a href="#">set_price(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	85
5.31	<a href="#">ProductHelper Namespace Reference</a>	85
5.31.1	<a href="#">Detailed Description</a>	86
5.31.2	<a href="#">Function Documentation</a>	86
5.31.2.1	<a href="#">get_producer(EntitySystem &amp;, tdt::uint)</a>	86
5.31.2.2	<a href="#">set_producer(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	86
5.32	<a href="#">ProductionHelper Namespace Reference</a>	86
5.32.1	<a href="#">Detailed Description</a>	87
5.32.2	<a href="#">Function Documentation</a>	87
5.32.2.1	<a href="#">get_production_blueprint(EntitySystem &amp;, tdt::uint)</a>	87
5.32.2.2	<a href="#">get_production_cooldown(EntitySystem &amp;, tdt::uint)</a>	87
5.32.2.3	<a href="#">get_production_count(EntitySystem &amp;, tdt::uint)</a>	88
5.32.2.4	<a href="#">get_production_limit(EntitySystem &amp;, tdt::uint)</a>	88
5.32.2.5	<a href="#">get_production_progress(EntitySystem &amp;, tdt::uint)</a>	88
5.32.2.6	<a href="#">set_production_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</a>	88
5.32.2.7	<a href="#">set_production_cooldown(EntitySystem &amp;, tdt::uint, tdt::real)</a>	89
5.32.2.8	<a href="#">set_production_count(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	89
5.32.2.9	<a href="#">set_production_limit(EntitySystem &amp;, tdt::uint, tdt::uint)</a>	89
5.32.2.10	<a href="#">set_production_progress(EntitySystem &amp;, tdt::uint, tdt::real)</a>	89
5.33	<a href="#">SpellHelper Namespace Reference</a>	90
5.33.1	<a href="#">Detailed Description</a>	90
5.33.2	<a href="#">Function Documentation</a>	90

5.33.2.1	<code>advance_curr_time(EntitySystem &amp;, tdt::uint, tdt::real)</code>	90
5.33.2.2	<code>cast(EntitySystem &amp;, tdt::uint)</code>	91
5.33.2.3	<code>get_blueprint(EntitySystem &amp;, tdt::uint)</code>	91
5.33.2.4	<code>get_cooldown(EntitySystem &amp;, tdt::uint)</code>	91
5.33.2.5	<code>get_curr_time(EntitySystem &amp;, tdt::uint)</code>	92
5.33.2.6	<code>set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	92
5.33.2.7	<code>set_cooldown(EntitySystem &amp;, tdt::uint, tdt::real)</code>	92
5.33.2.8	<code>set_curr_time(EntitySystem &amp;, tdt::uint, tdt::real)</code>	92
5.34	StructureHelper Namespace Reference	93
5.34.1	Detailed Description	93
5.34.2	Function Documentation	93
5.34.2.1	<code>add_residence(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	93
5.34.2.2	<code>add_residences(EntitySystem &amp;, tdt::uint, const std::vector&lt; tdt::uint &gt; &amp;)</code>	93
5.34.2.3	<code>get_radius(EntitySystem &amp;, tdt::uint)</code>	94
5.34.2.4	<code>is_walk_through(EntitySystem &amp;, tdt::uint)</code>	94
5.34.2.5	<code>set_radius(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	94
5.34.2.6	<code>set_walk_through(EntitySystem &amp;, tdt::uint, bool)</code>	94
5.35	TaskHandlerHelper Namespace Reference	95
5.35.1	Detailed Description	95
5.35.2	Function Documentation	95
5.35.2.1	<code>add_possible_task(EntitySystem &amp;, tdt::uint, TASK_TYPE)</code>	95
5.35.2.2	<code>clear_task_queue(EntitySystem &amp;, tdt::uint)</code>	96
5.35.2.3	<code>delete_possible_task(EntitySystem &amp;, tdt::uint, TASK_TYPE)</code>	96
5.35.2.4	<code>get_blueprint(EntitySystem &amp;, tdt::uint)</code>	96
5.35.2.5	<code>get_task_queue(EntitySystem &amp;, tdt::uint)</code>	96
5.35.2.6	<code>set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	97
5.35.2.7	<code>task_possible(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	97
5.35.2.8	<code>task_possible(EntitySystem &amp;, tdt::uint, TASK_TYPE)</code>	97
5.36	TaskHelper Namespace Reference	97
5.36.1	Detailed Description	98

5.36.2	Function Documentation	98
5.36.2.1	add_task(EntitySystem &, tdt::uint, tdt::uint, bool=false)	98
5.36.2.2	cancel_task(EntitySystem &, tdt::uint)	98
5.36.2.3	create_task(EntitySystem &, tdt::uint, TASK_TYPE)	99
5.36.2.4	get_task_source(EntitySystem &, tdt::uint)	99
5.36.2.5	get_task_target(EntitySystem &, tdt::uint)	99
5.36.2.6	get_task_type(EntitySystem &, tdt::uint)	99
5.36.2.7	is_complete(EntitySystem &, tdt::uint)	100
5.36.2.8	set_complete(EntitySystem &, tdt::uint)	100
5.36.2.9	set_task_source(EntitySystem &, tdt::uint, tdt::uint)	100
5.36.2.10	set_task_target(EntitySystem &, tdt::uint, tdt::uint)	100
5.36.2.11	set_task_type(EntitySystem &, tdt::uint, TASK_TYPE)	101
5.37	tdt Namespace Reference	101
5.37.1	Detailed Description	101
5.38	TimeHelper Namespace Reference	101
5.38.1	Detailed Description	102
5.38.2	Function Documentation	102
5.38.2.1	advance_curr_time(EntitySystem &, tdt::uint, tdt::real)	102
5.38.2.2	get_curr_time(EntitySystem &, tdt::uint)	102
5.38.2.3	get_target(EntitySystem &, tdt::uint)	103
5.38.2.4	get_time_limit(EntitySystem &, tdt::uint)	103
5.38.2.5	get_type(EntitySystem &, tdt::uint)	103
5.38.2.6	max_curr_time(EntitySystem &, tdt::uint)	103
5.38.2.7	set_target(EntitySystem &, tdt::uint, tdt::uint)	104
5.38.2.8	set_time_limit(EntitySystem &, tdt::uint, tdt::real)	104
5.38.2.9	set_type(EntitySystem &, tdt::uint, TIME_EVENT)	104
5.39	TriggerHelper Namespace Reference	104
5.39.1	Detailed Description	105
5.39.2	Function Documentation	105
5.39.2.1	can_be_triggered_by(EntitySystem &, tdt::uint, tdt::uint)	105

5.39.2.2	<code>get_blueprint(EntitySystem &amp;, tdt::uint)</code>	105
5.39.2.3	<code>get_cooldown(EntitySystem &amp;, tdt::uint)</code>	106
5.39.2.4	<code>get_linked_entity(EntitySystem &amp;, tdt::uint)</code>	106
5.39.2.5	<code>get_radius(EntitySystem &amp;, tdt::uint)</code>	106
5.39.2.6	<code>reset_timer(EntitySystem &amp;, tdt::uint)</code>	106
5.39.2.7	<code>set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	107
5.39.2.8	<code>set_cooldown(EntitySystem &amp;, tdt::uint, tdt::real)</code>	107
5.39.2.9	<code>set_linked_entity(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	107
5.39.2.10	<code>set_radius(EntitySystem &amp;, tdt::uint, tdt::real)</code>	107
5.39.2.11	<code>trigger(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	108
5.40	UpgradeHelper Namespace Reference	108
5.40.1	Detailed Description	108
5.40.2	Function Documentation	109
5.40.2.1	<code>add_experience(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	109
5.40.2.2	<code>can_level_up(EntitySystem &amp;, tdt::uint)</code>	109
5.40.2.3	<code>get_blueprint(EntitySystem &amp;, tdt::uint)</code>	109
5.40.2.4	<code>get_exp_needed(EntitySystem &amp;, tdt::uint)</code>	109
5.40.2.5	<code>get_experience(EntitySystem &amp;, tdt::uint)</code>	110
5.40.2.6	<code>get_level(EntitySystem &amp;, tdt::uint)</code>	110
5.40.2.7	<code>get_level_cap(EntitySystem &amp;, tdt::uint)</code>	110
5.40.2.8	<code>set_blueprint(EntitySystem &amp;, tdt::uint, const std::string &amp;)</code>	110
5.40.2.9	<code>set_exp_needed(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	111
5.40.2.10	<code>set_experience(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	111
5.40.2.11	<code>set_level(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	111
5.40.2.12	<code>set_level_cap(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	111
5.40.2.13	<code>upgrade(EntitySystem &amp;, tdt::uint)</code>	112
5.41	util Namespace Reference	112
5.41.1	Detailed Description	113
5.41.2	Typedef Documentation	113
5.41.2.1	<code>DEFAULT_PATH_TYPE</code>	113



5.41.3	Function Documentation	113
5.41.3.1	abs(int)	113
5.41.3.2	get_enum_direction(EntitySystem &, tdt::uint, tdt::uint)	114
5.41.3.3	get_random(tdt::uint, tdt::uint)	114
5.41.3.4	pathfind(EntitySystem &ents, tdt::uint id, tdt::uint target, util::heuristic::HEURIS← TIC &heuristic, bool add_path=true, bool allow_destruction=true)	114
5.42	util::effect Namespace Reference	115
5.42.1	Detailed Description	115
5.43	util::heuristic Namespace Reference	115
5.43.1	Detailed Description	115
5.44	util::path_type Namespace Reference	116
5.44.1	Detailed Description	116
5.45	util::pathfinding Namespace Reference	116
5.45.1	Detailed Description	116
<b>6</b>	<b>Class Documentation</b>	<b>117</b>
6.1	util::pathfinding::A_STAR< PATH_TYPE > Struct Template Reference	117
6.1.1	Detailed Description	117
6.2	AIComponent Struct Reference	118
6.2.1	Detailed Description	118
6.3	AISystem Class Reference	118
6.3.1	Detailed Description	119
6.3.2	Constructor & Destructor Documentation	119
6.3.2.1	AISystem(EntitySystem &)	119
6.3.2.2	~AISystem()=default	119
6.3.3	Member Function Documentation	120
6.3.3.1	force_update()	120
6.3.3.2	get_update_period() const	120
6.3.3.3	set_update_period(tdt::real)	120
6.3.3.4	update(tdt::real) override	120
6.3.4	Member Data Documentation	120

6.3.4.1	entities_ . . . . .	120
6.3.4.2	update_timer_ . . . . .	121
6.4	AlignComponent Struct Reference . . . . .	121
6.4.1	Detailed Description . . . . .	121
6.5	AlignComponent::AlignState Struct Reference . . . . .	122
6.5.1	Detailed Description . . . . .	122
6.6	util::path_type::BEST_PATH Struct Reference . . . . .	122
6.6.1	Detailed Description . . . . .	122
6.7	BuilderWindow Class Reference . . . . .	122
6.7.1	Detailed Description . . . . .	123
6.7.2	Constructor & Destructor Documentation . . . . .	124
6.7.2.1	BuilderWindow() . . . . .	124
6.7.2.2	~BuilderWindow()=default . . . . .	124
6.7.3	Member Function Documentation . . . . .	124
6.7.3.1	build(int) . . . . .	124
6.7.3.2	clear_buildings() . . . . .	124
6.7.3.3	dec_selection() . . . . .	124
6.7.3.4	get_building_(std::size_t) . . . . .	124
6.7.3.5	get_buildings() const . . . . .	125
6.7.3.6	inc_selection() . . . . .	125
6.7.3.7	init_() override . . . . .	125
6.7.3.8	register_building(const std::string &) . . . . .	125
6.7.3.9	set_placer(EntityPlacer *) . . . . .	125
6.7.3.10	update_selection_() . . . . .	126
6.7.4	Member Data Documentation . . . . .	126
6.7.4.1	buildings_ . . . . .	126
6.7.4.2	placer_ . . . . .	126
6.7.4.3	selection_number_ . . . . .	126
6.8	Camera Class Reference . . . . .	126
6.8.1	Detailed Description . . . . .	128

6.8.2	Constructor & Destructor Documentation . . . . .	128
6.8.2.1	Camera()=default . . . . .	128
6.8.2.2	~Camera()=default . . . . .	128
6.8.3	Member Function Documentation . . . . .	128
6.8.3.1	get_direction() const . . . . .	128
6.8.3.2	get_free_mode() const . . . . .	128
6.8.3.3	get_height() const . . . . .	129
6.8.3.4	get_position() const . . . . .	129
6.8.3.5	init(Ogre::Camera *) . . . . .	129
6.8.3.6	key_pressed(CEGUI::Key::Scan) . . . . .	129
6.8.3.7	key_released(CEGUI::Key::Scan) . . . . .	129
6.8.3.8	look_at(const Ogre::Vector2 &) . . . . .	129
6.8.3.9	move(DIRECTION::VAL, tdt::real) . . . . .	130
6.8.3.10	pitch(const Ogre::Degree &) . . . . .	130
6.8.3.11	reset() . . . . .	130
6.8.3.12	set_direction(const Ogre::Vector3 &) . . . . .	130
6.8.3.13	set_free_mode(bool) . . . . .	131
6.8.3.14	set_height(tdt::real) . . . . .	131
6.8.3.15	set_position(const Ogre::Vector2 &) . . . . .	131
6.8.3.16	set_start(const Ogre::Vector2 &, const Ogre::Vector2 &, tdt::real) . . . . .	131
6.8.3.17	update(tdt::real) . . . . .	132
6.8.3.18	yaw(const Ogre::Degree &) . . . . .	132
6.8.4	Member Data Documentation . . . . .	132
6.8.4.1	camera_ . . . . .	132
6.8.4.2	free_mode_ . . . . .	132
6.8.4.3	height_ . . . . .	132
6.8.4.4	movement_direction_ . . . . .	132
6.8.4.5	speed_ . . . . .	133
6.8.4.6	start_ . . . . .	133
6.9	CombatComponent Struct Reference . . . . .	133

6.9.1 Detailed Description . . . . .	134
6.10 CombatSystem Class Reference . . . . .	134
6.10.1 Detailed Description . . . . .	136
6.10.2 Constructor & Destructor Documentation . . . . .	136
6.10.2.1 CombatSystem(EntitySystem &, Ogre::SceneManager &, GridSystem &) . . . . .	136
6.10.2.2 ~CombatSystem()=default . . . . .	136
6.10.3 Member Function Documentation . . . . .	136
6.10.3.1 apply_damage_to_entities_in_range(tdt::uint, tdt::real, tdt::uint, tdt::uint) . . . . .	136
6.10.3.2 apply_effect_to_entities_in_range(tdt::uint id, COND &condition, EFFECT &effect, tdt::real range) . . . . .	137
6.10.3.3 apply_freeze_to(tdt::uint, tdt::real) . . . . .	137
6.10.3.4 apply_freeze_to_entities_in_range(tdt::uint, tdt::real, tdt::real) . . . . .	137
6.10.3.5 apply_heal_to_entities_in_range(tdt::uint, tdt::real) . . . . .	138
6.10.3.6 apply_slow_to(tdt::uint, tdt::real) . . . . .	138
6.10.3.7 apply_slow_to_entities_in_range(tdt::uint, tdt::real, tdt::real) . . . . .	138
6.10.3.8 create_homing_projectile(tdt::uint, CombatComponent &) . . . . .	138
6.10.3.9 enemy_in_range(tdt::uint) . . . . .	139
6.10.3.10 get_closest_entity(tdt::uint, bool=true, bool=false) const . . . . .	139
6.10.3.11 get_closest_entity(tdt::uint id, COND &condition, bool only_sight=true) const . . . . .	139
6.10.3.12 get_closest_entity_thats_not(tdt::uint, tdt::uint, bool=true, bool=false) const . . . . .	139
6.10.3.13 get_closest_gold_deposit(tdt::uint, bool=false) const . . . . .	140
6.10.3.14 get_closest_gold_vault(tdt::uint, bool=false, bool=false) const . . . . .	140
6.10.3.15 get_closest_structure(tdt::uint, bool=true, bool=false) const . . . . .	140
6.10.3.16 get_container() const . . . . .	141
6.10.3.17 get_container() const . . . . .	141
6.10.3.18 get_max_run_away_attempts() . . . . .	141
6.10.3.19 in_sight(tdt::uint, tdt::uint) const . . . . .	141
6.10.3.20 in_sight_wrt_BB(tdt::uint, tdt::uint) const . . . . .	141
6.10.3.21 run_away_from(tdt::uint, tdt::uint, tdt::uint) . . . . .	142
6.10.3.22 run_away_from_(tdt::uint, tdt::uint, tdt::uint) . . . . .	142
6.10.3.23 set_max_run_away_attempts(tdt::uint) . . . . .	142

6.10.3.24	<code>update(tdt::real) override</code>	143
6.10.4	Member Data Documentation	143
6.10.4.1	<code>entities_</code>	143
6.10.4.2	<code>grid_</code>	143
6.10.4.3	<code>max_run_away_attempts_</code>	143
6.10.4.4	<code>ray_caster_</code>	143
6.10.4.5	<code>ray_query_</code>	143
6.10.4.6	<code>run_away_queue_</code>	144
6.11	CommandComponent Struct Reference	144
6.11.1	Detailed Description	144
6.12	Component Struct Reference	145
6.12.1	Detailed Description	145
6.13	Console Class Reference	145
6.13.1	Detailed Description	146
6.13.2	Constructor & Destructor Documentation	146
6.13.2.1	<code>Console()</code>	146
6.13.2.2	<code>~Console()</code> =default	146
6.13.3	Member Function Documentation	146
6.13.3.1	<code>clear()</code>	146
6.13.3.2	<code>execute(const CEGUI::EventArgs &amp;)</code>	146
6.13.3.3	<code>get_history() const</code>	147
6.13.3.4	<code>handle_text(const CEGUI::EventArgs &amp;)</code>	147
6.13.3.5	<code>init_()</code> override	147
6.13.3.6	<code>print_text(const std::string &amp;, CEGUI::Colour=CEGUI::Colour{0xFFFFFFFF})</code>	147
6.13.3.7	<code>scroll_down(tdt::uint=1)</code>	147
6.13.3.8	<code>set_history(tdt::uint)</code>	148
6.13.3.9	<code>set_visible(bool)</code> override	148
6.13.3.10	<code>update_fps(tdt::real, tdt::real)</code>	148
6.13.4	Member Data Documentation	148
6.13.4.1	<code>console_history_</code>	148

6.13.4.2	<code>curr_command_</code>	149
6.13.4.3	<code>list_box_</code>	149
6.13.4.4	<code>time_since_last_fps_update_</code>	149
6.14	ConstructorComponent Struct Reference	149
6.14.1	Detailed Description	150
6.15	CounterComponent Struct Reference	150
6.15.1	Detailed Description	150
6.16	<code>util::effect::DAMAGE_EFFECT</code> Struct Reference	151
6.16.1	Detailed Description	151
6.16.2	Constructor & Destructor Documentation	151
6.16.2.1	<code>DAMAGE_EFFECT(EntitySystem &amp;, tdt::uint, tdt::uint)</code>	151
6.16.2.2	<code>~DAMAGE_EFFECT()</code> =default	151
6.16.3	Member Function Documentation	152
6.16.3.1	<code>operator()(tdt::uint)</code>	152
6.16.4	Member Data Documentation	152
6.16.4.1	<code>entities_</code>	152
6.16.4.2	<code>min_</code>	152
6.17	DestructorComponent Struct Reference	152
6.17.1	Detailed Description	153
6.18	EntityCreator Class Reference	153
6.18.1	Detailed Description	154
6.18.2	Constructor & Destructor Documentation	154
6.18.2.1	<code>EntityCreator(EntityPlacer &amp;, EntitySystem &amp;)</code>	154
6.18.2.2	<code>~EntityCreator()</code>	154
6.18.3	Member Function Documentation	154
6.18.3.1	<code>actualize_list(const CEGUI::EventArgs &amp;)</code>	154
6.18.3.2	<code>change_to_create(const CEGUI::EventArgs &amp;)</code>	155
6.18.3.3	<code>change_to_place(const CEGUI::EventArgs &amp;)</code>	155
6.18.3.4	<code>init_()</code> override	155
6.18.3.5	<code>place(const CEGUI::EventArgs &amp;)</code>	155

6.18.4	Member Data Documentation . . . . .	156
6.18.4.1	list_box_ . . . . .	156
6.18.4.2	placer_ . . . . .	156
6.18.4.3	registered_entities_ . . . . .	156
6.19	util::EntityDestroyer Class Reference . . . . .	156
6.19.1	Detailed Description . . . . .	156
6.19.2	Member Function Documentation . . . . .	156
6.19.2.1	destroy(EntitySystem &, tdt::uint) . . . . .	156
6.20	EntityPlacer Class Reference . . . . .	157
6.20.1	Detailed Description . . . . .	158
6.20.2	Constructor & Destructor Documentation . . . . .	158
6.20.2.1	EntityPlacer(EntitySystem &, GridSystem &, Ogre::SceneManager &) . . . . .	158
6.20.2.2	~EntityPlacer() . . . . .	159
6.20.3	Member Function Documentation . . . . .	159
6.20.3.1	can_place_when_game_paused() const . . . . .	159
6.20.3.2	is_visible() const . . . . .	159
6.20.3.3	place() . . . . .	159
6.20.3.4	set_current_entity_table(const std::string &, bool=false) . . . . .	159
6.20.3.5	set_visible(bool) . . . . .	159
6.20.3.6	toggle_placing_when_game_paused() . . . . .	160
6.20.3.7	update_position(const Ogre::Vector3 &) . . . . .	160
6.20.4	Member Data Documentation . . . . .	160
6.20.4.1	can_place_when_game_paused_ . . . . .	160
6.20.4.2	curr_position_ . . . . .	160
6.20.4.3	ent_ . . . . .	160
6.20.4.4	entities_ . . . . .	161
6.20.4.5	grid_ . . . . .	161
6.20.4.6	half_height_ . . . . .	161
6.20.4.7	mgr_ . . . . .	161
6.20.4.8	placing_node_ . . . . .	161

6.20.4.9	placing_structure_ . . . . .	161
6.20.4.10	price_ . . . . .	161
6.20.4.11	structure_radius_ . . . . .	162
6.20.4.12	table_name_ . . . . .	162
6.20.4.13	visible_ . . . . .	162
6.21	EntitySystem Class Reference . . . . .	162
6.21.1	Detailed Description . . . . .	169
6.21.2	Constructor & Destructor Documentation . . . . .	169
6.21.2.1	EntitySystem(Ogre::SceneManager &) . . . . .	169
6.21.2.2	~EntitySystem()=default . . . . .	169
6.21.3	Member Function Documentation . . . . .	169
6.21.3.1	add_component(tdt::uint id) . . . . .	169
6.21.3.2	add_component(tdt::uint, int) . . . . .	169
6.21.3.3	clean_up_component(tdt::uint) . . . . .	170
6.21.3.4	clean_up_component(tdt::uint id) . . . . .	170
6.21.3.5	clean_up_component(tdt::uint id) . . . . .	170
6.21.3.6	clean_up_component(tdt::uint id) . . . . .	170
6.21.3.7	cleanup() . . . . .	170
6.21.3.8	create_entity(const std::string &="","", const Ogre::Vector3 &=Ogre::Vector3{0.↵ f, 0.f, 0.f}) . . . . .	171
6.21.3.9	delete_component(tdt::uint id) . . . . .	171
6.21.3.10	delete_component(tdt::uint, int) . . . . .	171
6.21.3.11	delete_component_now(tdt::uint id) . . . . .	171
6.21.3.12	delete_component_now(tdt::uint, int) . . . . .	172
6.21.3.13	delete_entities() . . . . .	172
6.21.3.14	destroy_entity(tdt::uint) . . . . .	172
6.21.3.15	exists(tdt::uint) const . . . . .	172
6.21.3.16	get_component(tdt::uint id) . . . . .	172
6.21.3.17	get_component_container() . . . . .	173
6.21.3.18	get_component_container() . . . . .	173
6.21.3.19	get_component_list() const . . . . .	173



6.21.3.20	<code>get_new_id()</code>	173
6.21.3.21	<code>get_registered_entities()</code>	173
6.21.3.22	<code>get_scene_manager()</code>	173
6.21.3.23	<code>has_component(tdt::uint id)</code>	173
6.21.3.24	<code>has_component(tdt::uint, tdt::uint) const</code>	174
6.21.3.25	<code>init_function_arrays()</code>	174
6.21.3.26	<code>load_component(tdt::uint id, const std::string &amp;table_name)</code>	174
6.21.3.27	<code>load_component(tdt::uint id, const std::string &amp;table_name)</code>	174
6.21.3.28	<code>register_entity(const std::string &amp;)</code>	175
6.21.3.29	<code>set_component(tdt::uint id, COMP comp)</code>	175
6.21.3.30	<code>update(tdt::real) override</code>	175
6.21.4	Member Data Documentation	175
6.21.4.1	<code>constructors_to_be_called_</code>	175
6.21.4.2	<code>curr_id_</code>	176
6.21.4.3	<code>entities_</code>	176
6.21.4.4	<code>entity_register_</code>	176
6.21.4.5	<code>FACTION_NAME</code>	176
6.21.4.6	<code>loaders_</code>	176
6.21.4.7	<code>NO_BLUEPRINT</code>	176
6.21.4.8	<code>physics_</code>	176
6.21.4.9	<code>scene_</code>	177
6.21.4.10	<code>to_be_destroyed_</code>	177
6.22	EntityTracker Class Reference	177
6.22.1	Detailed Description	178
6.22.2	Constructor & Destructor Documentation	178
6.22.2.1	<code>EntityTracker()</code>	178
6.22.2.2	<code>~EntityTracker()=default</code>	178
6.22.3	Member Function Documentation	178
6.22.3.1	<code>clear()</code>	178
6.22.3.2	<code>get_tracked_entity() const</code>	178

6.22.3.3	<code>init_()</code> override	179
6.22.3.4	<code>init_upgrade_buttr(EntitySystem *)</code>	179
6.22.3.5	<code>set_tracked_entity(tdt::uint, EntitySystem &amp;)</code>	180
6.22.3.6	<code>show_upgrade_buttr(bool)</code>	180
6.22.3.7	<code>update_tracking(const std::string &amp;, const std::string &amp;)</code>	180
6.22.4	Member Data Documentation	180
6.22.4.1	<code>curr_tracked_entity_</code>	181
6.22.4.2	<code>entities_</code>	181
6.23	EventComponent Struct Reference	181
6.23.1	Detailed Description	182
6.24	EventHandlerComponent Struct Reference	182
6.24.1	Detailed Description	182
6.25	EventSystem Class Reference	183
6.25.1	Detailed Description	183
6.25.2	Constructor & Destructor Documentation	183
6.25.2.1	<code>EventSystem(EntitySystem &amp;)</code>	183
6.25.2.2	<code>~EventSystem()</code> =default	184
6.25.3	Member Function Documentation	184
6.25.3.1	<code>get_update_period()</code> const	184
6.25.3.2	<code>get_update_time_multiplier()</code> const	184
6.25.3.3	<code>handle_event_(tdt::uint, tdt::uint)</code>	184
6.25.3.4	<code>set_update_period(tdt::real)</code>	184
6.25.3.5	<code>set_update_time_multiplier(tdt::real)</code>	185
6.25.3.6	<code>update(tdt::real)</code> override	185
6.25.4	Member Data Documentation	185
6.25.4.1	<code>entities_</code>	185
6.25.4.2	<code>update_period_</code>	185
6.25.4.3	<code>update_time_multiplier_</code>	185
6.26	<code>lpp::Exception</code> Class Reference	186
6.26.1	Detailed Description	186

6.26.2	Constructor & Destructor Documentation . . . . .	186
6.26.2.1	Exception(const std::string &msg=""NO MSG"", Script::state L=nullptr) . . . . .	186
6.26.3	Member Function Documentation . . . . .	186
6.26.3.1	has_lua_state() const . . . . .	187
6.26.3.2	what() const . . . . .	187
6.26.3.3	what_lua() const . . . . .	187
6.26.4	Member Data Documentation . . . . .	187
6.26.4.1	L_ . . . . .	187
6.26.4.2	msg_ . . . . .	187
6.27	ExperienceValueComponent Struct Reference . . . . .	187
6.27.1	Detailed Description . . . . .	188
6.28	ExplosionComponent Struct Reference . . . . .	188
6.28.1	Detailed Description . . . . .	189
6.29	FactionComponent Struct Reference . . . . .	189
6.29.1	Detailed Description . . . . .	189
6.30	util::path_type::FIRST_PATH Struct Reference . . . . .	190
6.30.1	Detailed Description . . . . .	190
6.31	util::effect::FREEZE_EFFECT Struct Reference . . . . .	190
6.31.1	Detailed Description . . . . .	190
6.31.2	Constructor & Destructor Documentation . . . . .	190
6.31.2.1	FREEZE_EFFECT(EntitySystem &, tdt::real) . . . . .	190
6.31.2.2	~FREEZE_EFFECT()=default . . . . .	191
6.31.3	Member Function Documentation . . . . .	191
6.31.3.1	operator()(tdt::uint) . . . . .	191
6.31.4	Member Data Documentation . . . . .	191
6.31.4.1	entities_ . . . . .	191
6.31.4.2	time_ . . . . .	191
6.32	Game Class Reference . . . . .	192
6.32.1	Detailed Description . . . . .	194
6.32.2	Constructor & Destructor Documentation . . . . .	194

6.32.2.1	<code>Game()</code>	194
6.32.2.2	<code>~Game()</code>	194
6.32.3	Member Function Documentation	194
6.32.3.1	<code>create_empty_level(tdt::uint, tdt::uint)</code>	194
6.32.3.2	<code>frameRenderingQueued(const Ogre::FrameEvent &amp;) override</code>	195
6.32.3.3	<code>get_mouse_click_position(const OIS::MouseEvent &amp;) const</code>	195
6.32.3.4	<code>get_throne_id() const</code>	195
6.32.3.5	<code>keyPressed(const OIS::KeyEvent &amp;) override</code>	195
6.32.3.6	<code>keyReleased(const OIS::KeyEvent &amp;) override</code>	195
6.32.3.7	<code>mouseMoved(const OIS::MouseEvent &amp;) override</code>	195
6.32.3.8	<code>mousePressed(const OIS::MouseEvent &amp;, OIS::MouseButtonID) override</code>	196
6.32.3.9	<code>mouseReleased(const OIS::MouseEvent &amp;, OIS::MouseButtonID) override</code>	196
6.32.3.10	<code>new_game(tdt::uint, tdt::uint)</code>	196
6.32.3.11	<code>ogre_init()</code>	196
6.32.3.12	<code>ois_to_cegui(OIS::MouseButtonID)</code>	196
6.32.3.13	<code>reset_camera()</code>	197
6.32.3.14	<code>reset_unlocks()</code>	197
6.32.3.15	<code>run()</code>	197
6.32.3.16	<code>set_state(GAME_STATE)</code>	197
6.32.3.17	<code>set_throne_id(tdt::uint)</code>	197
6.32.3.18	<code>toggle_camera_free_mode()</code>	197
6.32.3.19	<code>update(tdt::real)</code>	198
6.32.3.20	<code>windowClosed(Ogre::RenderWindow *rw) override</code>	199
6.32.3.21	<code>windowResized(Ogre::RenderWindow *rw) override</code>	199
6.32.4	Member Data Documentation	199
6.32.4.1	<code>entity_creator_</code>	199
6.32.4.2	<code>entity_system_</code>	199
6.32.4.3	<code>game_serializer_</code>	199
6.32.4.4	<code>ground_</code>	199
6.32.4.5	<code>ground_entity_</code>	200

6.32.4.6	level_generator_ . . . . .	200
6.32.4.7	main_cam_ . . . . .	200
6.32.4.8	mouse_position_ . . . . .	200
6.32.4.9	placer_ . . . . .	200
6.32.4.10	renderer_ . . . . .	200
6.32.4.11	root_ . . . . .	200
6.32.4.12	selection_box_ . . . . .	201
6.32.4.13	spell_caster_ . . . . .	201
6.32.4.14	state_ . . . . .	201
6.32.4.15	systems_ . . . . .	201
6.32.4.16	throne_id_ . . . . .	201
6.33	GameLog Class Reference . . . . .	201
6.33.1	Detailed Description . . . . .	202
6.33.2	Constructor & Destructor Documentation . . . . .	202
6.33.2.1	GameLog() . . . . .	202
6.33.2.2	~GameLog()=default . . . . .	203
6.33.3	Member Function Documentation . . . . .	203
6.33.3.1	clear() . . . . .	203
6.33.3.2	get_history() const . . . . .	203
6.33.3.3	init_() override . . . . .	203
6.33.3.4	print(const std::string &) . . . . .	203
6.33.3.5	set_history(tdt::uint) . . . . .	203
6.33.4	Member Data Documentation . . . . .	204
6.33.4.1	log_ . . . . .	204
6.33.4.2	log_history_ . . . . .	204
6.34	GameSerializer Class Reference . . . . .	204
6.34.1	Detailed Description . . . . .	207
6.34.2	Constructor & Destructor Documentation . . . . .	207
6.34.2.1	GameSerializer(EntitySystem &) . . . . .	207
6.34.2.2	~GameSerializer() . . . . .	207

6.34.3	Member Function Documentation	207
6.34.3.1	load_game(Game &, const std::string &=""quick_save"")	207
6.34.3.2	save_component(tdt::uint, const std::string &)	207
6.34.3.3	save_game(Game &, const std::string &=""quick_save"")	208
6.34.3.4	save_tasks()	208
6.34.3.5	save_unlocks()	208
6.34.3.6	save_wave_system(Game &)	208
6.34.4	Member Data Documentation	208
6.34.4.1	entities_	208
6.34.4.2	file_	208
6.34.4.3	save_entities_	209
6.34.4.4	script_	209
6.34.4.5	serializers_	209
6.34.4.6	task_pairs_	209
6.35	GoldComponent Struct Reference	209
6.35.1	Detailed Description	210
6.36	GraphicsComponent Struct Reference	210
6.36.1	Detailed Description	211
6.37	GraphicsSystem Class Reference	211
6.37.1	Detailed Description	211
6.37.2	Constructor & Destructor Documentation	211
6.37.2.1	GraphicsSystem(EntitySystem &)	211
6.37.2.2	~GraphicsSystem()=default	212
6.37.3	Member Function Documentation	212
6.37.3.1	get_update_period() const	212
6.37.3.2	set_update_period(tdt::real)	212
6.37.3.3	update(tdt::real) override	212
6.37.4	Member Data Documentation	212
6.37.4.1	entities_	212
6.37.4.2	update_timer_	213

6.38 Grid Class Reference . . . . .	213
6.38.1 Detailed Description . . . . .	215
6.38.2 Constructor & Destructor Documentation . . . . .	215
6.38.2.1 Grid(const Grid &)=delete . . . . .	215
6.38.2.2 Grid()=default . . . . .	215
6.38.2.3 ~Grid() . . . . .	215
6.38.3 Member Function Documentation . . . . .	215
6.38.3.1 add_freed(tdt::uint) . . . . .	215
6.38.3.2 add_node(EntitySystem &, Ogre::Vector2) . . . . .	215
6.38.3.3 add_unfreed(tdt::uint) . . . . .	216
6.38.3.4 clear_freed() . . . . .	216
6.38.3.5 clear_unfreed() . . . . .	216
6.38.3.6 create_graph(EntitySystem &, Ogre::Vector2, tdt::uint, tdt::uint, tdt::real) . . . . .	216
6.38.3.7 distribute_to_adjacent_free_nodes(EntitySystem &, tdt::uint, const std::vector<tdt::uint > &) . . . . .	216
6.38.3.8 get_center_position(EntitySystem &) const . . . . .	217
6.38.3.9 get_distance() const . . . . .	217
6.38.3.10 get_freed() const . . . . .	217
6.38.3.11 get_node(tdt::uint, tdt::uint) const . . . . .	217
6.38.3.12 get_node_from_position(tdt::real, tdt::real) const . . . . .	217
6.38.3.13 get_random_free_node() const . . . . .	218
6.38.3.14 get_unfreed() const . . . . .	218
6.38.3.15 in_board(tdt::uint) const . . . . .	218
6.38.3.16 instance() . . . . .	218
6.38.3.17 link_(tdt::uint, std::vector< GridNodeComponent * > &) . . . . .	218
6.38.3.18 place_at_random_free_node(EntitySystem &, tdt::uint) . . . . .	219
6.38.3.19 remove_node(tdt::uint) . . . . .	219
6.38.4 Friends And Related Function Documentation . . . . .	219
6.38.4.1 GameSerializer . . . . .	219
6.38.5 Member Data Documentation . . . . .	219
6.38.5.1 distance_ . . . . .	219

6.38.5.2	<a href="#">free_nodes_</a>	220
6.38.5.3	<a href="#">freed_</a>	220
6.38.5.4	<a href="#">nodes_</a>	220
6.38.5.5	<a href="#">start_</a>	220
6.38.5.6	<a href="#">starting_index_</a>	220
6.38.5.7	<a href="#">width_</a>	220
6.39	<a href="#">GridNodeComponent Struct Reference</a>	221
6.39.1	<a href="#">Detailed Description</a>	221
6.40	<a href="#">GridSystem Class Reference</a>	222
6.40.1	<a href="#">Detailed Description</a>	223
6.40.2	<a href="#">Constructor &amp; Destructor Documentation</a>	223
6.40.2.1	<a href="#">GridSystem(EntitySystem &amp;, Ogre::SceneManager &amp;)</a>	223
6.40.2.2	<a href="#">~GridSystem()=default</a>	223
6.40.3	<a href="#">Member Function Documentation</a>	223
6.40.3.1	<a href="#">create_graphics()</a>	223
6.40.3.2	<a href="#">delete_graphics()</a>	223
6.40.3.3	<a href="#">is_visible() const</a>	223
6.40.3.4	<a href="#">place_structure(tdt::uint, tdt::uint, tdt::uint)</a>	223
6.40.3.5	<a href="#">set_visible(bool)</a>	224
6.40.3.6	<a href="#">update(tdt::real) override</a>	224
6.40.3.7	<a href="#">update_neighbours_(tdt::uint)</a>	224
6.40.4	<a href="#">Member Data Documentation</a>	224
6.40.4.1	<a href="#">entities_</a>	224
6.40.4.2	<a href="#">graphics_loaded_</a>	225
6.40.4.3	<a href="#">scene_mgr_</a>	225
6.41	<a href="#">GUI Class Reference</a>	225
6.41.1	<a href="#">Detailed Description</a>	227
6.41.2	<a href="#">Constructor &amp; Destructor Documentation</a>	227
6.41.2.1	<a href="#">~GUI()</a>	227
6.41.2.2	<a href="#">GUI(const GUI &amp;)=delete</a>	227



6.41.2.3	GUI()	228
6.41.3	Member Function Documentation	228
6.41.3.1	escape_pressed()	228
6.41.3.2	get_builder()	228
6.41.3.3	get_console()	228
6.41.3.4	get_curr_tool()	228
6.41.3.5	get_log()	228
6.41.3.6	get_message()	229
6.41.3.7	get_options()	229
6.41.3.8	get_research()	229
6.41.3.9	get_spell_casting()	229
6.41.3.10	get_top_bar()	229
6.41.3.11	get_tracker()	229
6.41.3.12	get_window()	229
6.41.3.13	get_window(const std::string &)	229
6.41.3.14	init(Game *)	230
6.41.3.15	instance()	230
6.41.3.16	is_visible() const	230
6.41.3.17	is_visible(const std::string &) const	230
6.41.3.18	list_directory(const std::string &, CEGUI::Listbox &, bool=false)	231
6.41.3.19	set_curr_tool(const std::string &)	231
6.41.3.20	set_curr_tool_visible(bool)	231
6.41.3.21	set_visible(bool)	231
6.41.3.22	set_visible(const std::string &, bool)	231
6.41.3.23	show_load_save_dialog(const std::string &)	232
6.41.4	Member Data Documentation	232
6.41.4.1	builder_	232
6.41.4.2	console_	232
6.41.4.3	curr_tool_	232
6.41.4.4	game_	232

6.41.4.5	log_ . . . . .	233
6.41.4.6	menu_ . . . . .	233
6.41.4.7	message_ . . . . .	233
6.41.4.8	options_ . . . . .	233
6.41.4.9	research_ . . . . .	233
6.41.4.10	spell_casting_ . . . . .	233
6.41.4.11	top_bar_ . . . . .	233
6.41.4.12	tracker_ . . . . .	233
6.41.4.13	window_ . . . . .	234
6.42	GUIWindow Class Reference . . . . .	234
6.42.1	Detailed Description . . . . .	235
6.42.2	Constructor & Destructor Documentation . . . . .	235
6.42.2.1	GUIWindow() . . . . .	235
6.42.2.2	~GUIWindow()=default . . . . .	235
6.42.3	Member Function Documentation . . . . .	236
6.42.3.1	init(CEGUI::Window *) . . . . .	236
6.42.3.2	init_()=0 . . . . .	236
6.42.3.3	is_visible() const . . . . .	236
6.42.3.4	set_visible(bool) . . . . .	236
6.42.4	Member Data Documentation . . . . .	236
6.42.4.1	window_ . . . . .	236
6.43	util::HAS_GOLD Struct Reference . . . . .	237
6.43.1	Detailed Description . . . . .	237
6.43.2	Constructor & Destructor Documentation . . . . .	237
6.43.2.1	HAS_GOLD(EntitySystem &) . . . . .	237
6.43.2.2	~HAS_GOLD()=default . . . . .	237
6.43.3	Member Function Documentation . . . . .	238
6.43.3.1	operator()(tdt::uint) . . . . .	238
6.43.4	Member Data Documentation . . . . .	238
6.43.4.1	entities_ . . . . .	238

6.44	util::effect::HEAL_EFFECT Struct Reference . . . . .	238
6.44.1	Detailed Description . . . . .	239
6.44.2	Constructor & Destructor Documentation . . . . .	239
6.44.2.1	HEAL_EFFECT(EntitySystem &) . . . . .	239
6.44.2.2	~HEAL_EFFECT()=default . . . . .	239
6.44.3	Member Function Documentation . . . . .	239
6.44.3.1	operator()(tdt::uint) . . . . .	239
6.44.4	Member Data Documentation . . . . .	239
6.44.4.1	entities_ . . . . .	239
6.45	HealthComponent Struct Reference . . . . .	240
6.45.1	Detailed Description . . . . .	240
6.46	HealthSystem Class Reference . . . . .	240
6.46.1	Detailed Description . . . . .	241
6.46.2	Constructor & Destructor Documentation . . . . .	241
6.46.2.1	HealthSystem(EntitySystem &) . . . . .	241
6.46.2.2	~HealthSystem()=default . . . . .	242
6.46.3	Member Function Documentation . . . . .	242
6.46.3.1	get_regen_period() const . . . . .	242
6.46.3.2	set_regen_period(tdt::real) . . . . .	242
6.46.3.3	update(tdt::real) override . . . . .	242
6.46.3.4	update_regen(tdt::real) . . . . .	242
6.46.4	Member Data Documentation . . . . .	243
6.46.4.1	entities_ . . . . .	243
6.46.4.2	regen_ . . . . .	243
6.46.4.3	regen_period_ . . . . .	243
6.46.4.4	regen_timer_ . . . . .	243
6.47	util::heuristic::HEURISTIC Struct Reference . . . . .	243
6.47.1	Detailed Description . . . . .	244
6.48	HomingComponent Struct Reference . . . . .	244
6.48.1	Detailed Description . . . . .	245

6.49	InputComponent Struct Reference	245
6.49.1	Detailed Description	245
6.50	InputSystem Class Reference	246
6.50.1	Detailed Description	247
6.50.2	Constructor & Destructor Documentation	247
6.50.2.1	InputSystem(EntitySystem &, OIS::Keyboard &, Ogre::Camera &)	247
6.50.2.2	~InputSystem()=default	247
6.50.3	Member Function Documentation	247
6.50.3.1	is_first_person() const	247
6.50.3.2	rebind(int, int)	247
6.50.3.3	set_first_person(bool, tdt::uint=0)	248
6.50.3.4	update(tdt::real) override	248
6.50.4	Member Data Documentation	248
6.50.4.1	ai_backup_	248
6.50.4.2	cam_	248
6.50.4.3	cam_position_	248
6.50.4.4	delete_input_	249
6.50.4.5	entities_	249
6.50.4.6	first_person_	249
6.50.4.7	first_person_id_	249
6.50.4.8	KEY_UP	249
6.50.4.9	keyboard_	249
6.50.4.10	task_backup_	249
6.51	util::IS_ENEMY Struct Reference	250
6.51.1	Detailed Description	250
6.51.2	Constructor & Destructor Documentation	250
6.51.2.1	IS_ENEMY(EntitySystem &, tdt::uint)	250
6.51.2.2	~IS_ENEMY()=default	250
6.51.3	Member Function Documentation	251
6.51.3.1	operator()(tdt::uint)	251

6.51.4	Member Data Documentation . . . . .	251
6.51.4.1	enemy_faction_ . . . . .	251
6.51.4.2	entities_ . . . . .	251
6.52	util::IS_FRIENDLY Struct Reference . . . . .	251
6.52.1	Detailed Description . . . . .	252
6.52.2	Constructor & Destructor Documentation . . . . .	252
6.52.2.1	IS_FRIENDLY(EntitySystem &, tdt::uint) . . . . .	252
6.52.2.2	~IS_FRIENDLY()=default . . . . .	252
6.52.3	Member Function Documentation . . . . .	252
6.52.3.1	operator()(tdt::uint) . . . . .	252
6.52.4	Member Data Documentation . . . . .	252
6.52.4.1	entities_ . . . . .	253
6.52.4.2	faction_ . . . . .	253
6.53	util::IS_FRIENDLY_OR_NEUTRAL Struct Reference . . . . .	253
6.53.1	Detailed Description . . . . .	253
6.53.2	Constructor & Destructor Documentation . . . . .	253
6.53.2.1	IS_FRIENDLY_OR_NEUTRAL(EntitySystem &, tdt::uint) . . . . .	253
6.53.2.2	~IS_FRIENDLY_OR_NEUTRAL()=default . . . . .	254
6.53.3	Member Function Documentation . . . . .	254
6.53.3.1	operator()(tdt::uint) . . . . .	254
6.53.4	Member Data Documentation . . . . .	254
6.53.4.1	entities_ . . . . .	254
6.53.4.2	faction_ . . . . .	254
6.54	util::IS_GOLD_VAULT Struct Reference . . . . .	255
6.54.1	Detailed Description . . . . .	255
6.54.2	Constructor & Destructor Documentation . . . . .	255
6.54.2.1	IS_GOLD_VAULT(EntitySystem &) . . . . .	255
6.54.2.2	~IS_GOLD_VAULT()=default . . . . .	255
6.54.3	Member Function Documentation . . . . .	256
6.54.3.1	operator()(tdt::uint) . . . . .	256

6.54.4	Member Data Documentation . . . . .	256
6.54.4.1	entities_ . . . . .	256
6.55	level_generators::LevelGenerator Class Reference . . . . .	256
6.55.1	Detailed Description . . . . .	257
6.55.2	Constructor & Destructor Documentation . . . . .	257
6.55.2.1	LevelGenerator(EntitySystem &, tdt::uint) . . . . .	257
6.55.2.2	~LevelGenerator()=default . . . . .	257
6.55.3	Member Function Documentation . . . . .	257
6.55.3.1	generate(tdt::uint, tdt::uint, WaveSystem &)=0 . . . . .	257
6.55.4	Member Data Documentation . . . . .	258
6.55.4.1	cycles_ . . . . .	258
6.55.4.2	entities_ . . . . .	258
6.56	LightComponent Struct Reference . . . . .	258
6.56.1	Detailed Description . . . . .	258
6.57	LimitedLifeSpanComponent Struct Reference . . . . .	259
6.57.1	Detailed Description . . . . .	259
6.58	util::effect::LOWER_SPEED_EFFECT Struct Reference . . . . .	259
6.58.1	Detailed Description . . . . .	260
6.58.2	Constructor & Destructor Documentation . . . . .	260
6.58.2.1	LOWER_SPEED_EFFECT(EntitySystem &, tdt::real) . . . . .	260
6.58.2.2	~LOWER_SPEED_EFFECT()=default . . . . .	260
6.58.3	Member Function Documentation . . . . .	260
6.58.3.1	operator()(tdt::uint) . . . . .	260
6.58.4	Member Data Documentation . . . . .	260
6.58.4.1	entities_ . . . . .	261
6.58.4.2	time_ . . . . .	261
6.59	LuaInterface Class Reference . . . . .	261
6.59.1	Detailed Description . . . . .	271
6.59.2	Constructor & Destructor Documentation . . . . .	271
6.59.2.1	LuaInterface()=delete . . . . .	271

6.59.3	Member Function Documentation	271
6.59.3.1	init(Game *)	271
6.59.3.2	lua_get_avg_fps(lpp::Script::state)	271
6.59.4	Member Data Documentation	271
6.59.4.1	lua_this	271
6.60	ManaComponent Struct Reference	272
6.60.1	Detailed Description	272
6.61	ManaCrystalComponent Struct Reference	272
6.61.1	Detailed Description	273
6.62	ManaSpellSystem Class Reference	273
6.62.1	Detailed Description	274
6.62.2	Constructor & Destructor Documentation	274
6.62.2.1	ManaSpellSystem(EntitySystem &)	274
6.62.2.2	~ManaSpellSystem()=default	274
6.62.3	Member Function Documentation	274
6.62.3.1	get_regen_period() const	274
6.62.3.2	set_regen_period(tdt::real)	274
6.62.3.3	update(tdt::real) override	275
6.62.4	Member Data Documentation	275
6.62.4.1	entities_	275
6.62.4.2	regen_timer_	275
6.63	util::heuristic::MANHATTAN_DISTANCE Struct Reference	275
6.63.1	Detailed Description	276
6.64	MessageToPlayerWindow Class Reference	276
6.64.1	Detailed Description	277
6.64.2	Constructor & Destructor Documentation	277
6.64.2.1	MessageToPlayerWindow()	277
6.64.2.2	~MessageToPlayerWindow()=default	277
6.64.3	Member Function Documentation	277
6.64.3.1	init_()	277

6.64.3.2	<code>reset_butt_labels()</code>	277
6.64.3.3	<code>set_butt_label(const std::string &amp;, const std::string &amp;)</code>	277
6.64.3.4	<code>show(const std::string &amp;=""NONE"", const std::string &amp;=""NONE"", const std::string &amp;=""NONE"", const std::string &amp;=""NONE"")</code>	278
6.64.4	Member Data Documentation	278
6.64.4.1	<code>ok_func_</code>	278
6.64.4.2	<code>script_</code>	278
6.65	MineComponent Struct Reference	279
6.65.1	Detailed Description	279
6.66	MovementComponent Struct Reference	279
6.66.1	Detailed Description	280
6.67	MovementSystem Class Reference	280
6.67.1	Detailed Description	281
6.67.2	Constructor & Destructor Documentation	281
6.67.2.1	<code>MovementSystem(EntitySystem &amp;)</code>	281
6.67.2.2	<code>~MovementSystem()</code>	281
6.67.3	Member Function Documentation	281
6.67.3.1	<code>can_move_to(std::size_t, Ogre::Vector3)</code>	281
6.67.3.2	<code>checked_move(std::size_t, Ogre::Vector3)</code>	281
6.67.3.3	<code>move(std::size_t, Ogre::Vector3)</code>	282
6.67.3.4	<code>update(Ogre::Real)</code>	282
6.67.4	Member Data Documentation	282
6.67.4.1	<code>entities_</code>	282
6.67.4.2	<code>last_delta_</code>	283
6.68	NameComponent Struct Reference	283
6.68.1	Detailed Description	283
6.69	<code>util::heuristic::NO_HEURISTIC</code> Struct Reference	284
6.69.1	Detailed Description	284
6.70	NotificationComponent Struct Reference	284
6.70.1	Detailed Description	285
6.71	OnHitComponent Struct Reference	285



6.71.1 Detailed Description . . . . .	286
6.72 OptionsWindow Class Reference . . . . .	286
6.72.1 Detailed Description . . . . .	287
6.72.2 Constructor & Destructor Documentation . . . . .	287
6.72.2.1 OptionsWindow() . . . . .	287
6.72.2.2 ~OptionsWindow()=default . . . . .	288
6.72.3 Member Function Documentation . . . . .	288
6.72.3.1 add_start_parameters(Ogre::RenderWindow *, Ogre::Viewport *, CEGUI::↔ OgreRenderer *) . . . . .	288
6.72.3.2 apply_() . . . . .	288
6.72.3.3 get_key_bind_name_(KEY_BIND_ACTION::VAL) . . . . .	288
6.72.3.4 init_() override . . . . .	288
6.72.3.5 key_pressed(CEGUI::Key::Scan) . . . . .	289
6.72.3.6 set_key_bind(KEY_BIND_ACTION::VAL, CEGUI::Key::Scan) . . . . .	289
6.72.3.7 update_labels_() . . . . .	289
6.72.4 Member Data Documentation . . . . .	289
6.72.4.1 actions_ . . . . .	289
6.72.4.2 currently_binded_action_ . . . . .	289
6.72.4.3 fullscreen_ . . . . .	289
6.72.4.4 key_binds_ . . . . .	290
6.72.4.5 key_names_ . . . . .	290
6.72.4.6 render_window_ . . . . .	290
6.72.4.7 renderer_ . . . . .	290
6.72.4.8 resolutions_ . . . . .	290
6.72.4.9 view_ . . . . .	290
6.72.4.10 width_ . . . . .	290
6.73 PathfindingComponent Struct Reference . . . . .	291
6.73.1 Detailed Description . . . . .	291
6.74 PhysicsComponent Struct Reference . . . . .	291
6.74.1 Detailed Description . . . . .	292
6.75 Player Class Reference . . . . .	292

6.75.1 Detailed Description . . . . .	294
6.75.2 Constructor & Destructor Documentation . . . . .	294
6.75.2.1 ~Player() . . . . .	294
6.75.2.2 Player() . . . . .	295
6.75.3 Member Function Documentation . . . . .	295
6.75.3.1 add_curr_unit(tdt::uint) . . . . .	295
6.75.3.2 add_gold(tdt::uint) . . . . .	295
6.75.3.3 add_mana(tdt::uint) . . . . .	295
6.75.3.4 add_mana_regen(tdt::uint) . . . . .	295
6.75.3.5 add_max_mana(tdt::uint) . . . . .	296
6.75.3.6 add_max_unit(tdt::uint) . . . . .	296
6.75.3.7 get_gold() const . . . . .	296
6.75.3.8 get_initial_buildings() const . . . . .	296
6.75.3.9 get_initial_spells() const . . . . .	296
6.75.3.10 get_mana() const . . . . .	296
6.75.3.11 get_mana_regen() const . . . . .	297
6.75.3.12 get_max_mana() const . . . . .	297
6.75.3.13 init(EntitySystem *) . . . . .	297
6.75.3.14 instance() . . . . .	297
6.75.3.15 nulify_all_stats() . . . . .	297
6.75.3.16 reset() . . . . .	297
6.75.3.17 set_initial_unlocks(const std::vector< std::string > &, const std::vector< std::string > &) . . . . .	297
6.75.3.18 sub_curr_unit(tdt::uint) . . . . .	298
6.75.3.19 sub_gold(tdt::uint) . . . . .	298
6.75.3.20 sub_mana(tdt::uint) . . . . .	298
6.75.3.21 sub_mana_regen(tdt::uint) . . . . .	298
6.75.3.22 sub_max_mana(tdt::uint) . . . . .	299
6.75.3.23 sub_max_unit(tdt::uint) . . . . .	299
6.75.4 Member Data Documentation . . . . .	299
6.75.4.1 entities_ . . . . .	299

6.75.4.2	gold_ . . . . .	299
6.75.4.3	initial_building_unlocks_ . . . . .	300
6.75.4.4	initial_spell_unlocks_ . . . . .	300
6.75.4.5	mana_ . . . . .	300
6.75.4.6	mana_regen_ . . . . .	300
6.75.4.7	max_mana_ . . . . .	300
6.75.4.8	uint_max_ . . . . .	300
6.75.4.9	units_curr_ . . . . .	300
6.75.4.10	units_max_ . . . . .	301
6.76	util::heuristic::PORTAL_HEURISTIC Struct Reference . . . . .	301
6.76.1	Detailed Description . . . . .	301
6.76.2	Member Function Documentation . . . . .	301
6.76.2.1	get_closest_portal(tdt::uint id) . . . . .	301
6.77	PortalComponent Struct Reference . . . . .	302
6.77.1	Detailed Description . . . . .	302
6.78	PriceComponent Struct Reference . . . . .	302
6.78.1	Detailed Description . . . . .	303
6.79	ProductComponent Struct Reference . . . . .	303
6.79.1	Detailed Description . . . . .	304
6.80	ProductionComponent Struct Reference . . . . .	304
6.80.1	Detailed Description . . . . .	304
6.81	ProductionSystem Class Reference . . . . .	305
6.81.1	Detailed Description . . . . .	305
6.81.2	Constructor & Destructor Documentation . . . . .	305
6.81.2.1	ProductionSystem(EntitySystem &) . . . . .	305
6.81.2.2	~ProductionSystem() . . . . .	306
6.81.3	Member Function Documentation . . . . .	306
6.81.3.1	get_time_multiplier() . . . . .	306
6.81.3.2	set_time_multiplier(tdt::real) . . . . .	306
6.81.3.3	spawn_entity(tdt::uint, const std::string &) . . . . .	306

6.81.3.4	<a href="#">update(tdt::real) override</a>	307
6.81.4	<a href="#">Member Data Documentation</a>	308
6.81.4.1	<a href="#">entities_</a>	308
6.81.4.2	<a href="#">grid_</a>	308
6.81.4.3	<a href="#">time_multiplier_</a>	308
6.82	<a href="#">util::path_type::RANDOM_PATH&lt; UPPER &gt; Struct Template Reference</a>	308
6.82.1	<a href="#">Detailed Description</a>	309
6.83	<a href="#">level_generators::RandomLevelGenerator Class Reference</a>	309
6.83.1	<a href="#">Detailed Description</a>	309
6.83.2	<a href="#">Constructor &amp; Destructor Documentation</a>	309
6.83.2.1	<a href="#">RandomLevelGenerator(EntitySystem &amp;, tdt::uint)</a>	309
6.83.3	<a href="#">Member Function Documentation</a>	310
6.83.3.1	<a href="#">generate(tdt::uint, tdt::uint, WaveSystem &amp;) override</a>	310
6.84	<a href="#">RayCaster Class Reference</a>	310
6.84.1	<a href="#">Detailed Description</a>	311
6.84.2	<a href="#">Constructor &amp; Destructor Documentation</a>	311
6.84.2.1	<a href="#">RayCaster(Ogre::SceneManager &amp;)</a>	311
6.84.3	<a href="#">Member Function Documentation</a>	311
6.84.3.1	<a href="#">cast(const Ogre::Vector3 &amp;, const Ogre::Vector3 &amp;, const std::string &amp;="") const</a>	311
6.84.3.2	<a href="#">get_info(const Ogre::Entity &amp;, tdt::uint &amp;, tdt::uint &amp;, std::vector&lt; Ogre::Vector3 &gt; &amp;, std::vector&lt; tdt::uint &gt; &amp;, const Ogre::Vector3 &amp;, const Ogre::Quaternion &amp;, const Ogre::Vector3 &amp;) const</a>	312
6.84.4	<a href="#">Member Data Documentation</a>	312
6.84.4.1	<a href="#">query_</a>	312
6.85	<a href="#">ResearchWindow Class Reference</a>	312
6.85.1	<a href="#">Detailed Description</a>	314
6.85.2	<a href="#">Constructor &amp; Destructor Documentation</a>	314
6.85.2.1	<a href="#">ResearchWindow()</a>	314
6.85.2.2	<a href="#">~ResearchWindow()=default</a>	314
6.85.3	<a href="#">Member Function Documentation</a>	314
6.85.3.1	<a href="#">dummy_unlock(tdt::uint, tdt::uint)</a>	314

6.85.3.2	<a href="#">free_research()</a>	314
6.85.3.3	<a href="#">get_price_(tdt::uint, tdt::uint)</a>	315
6.85.3.4	<a href="#">get_unlocked() const</a>	315
6.85.3.5	<a href="#">init_() override</a>	315
6.85.3.6	<a href="#">is_unlocked_(tdt::uint, tdt::uint)</a>	315
6.85.3.7	<a href="#">research_all()</a>	315
6.85.3.8	<a href="#">reset_research()</a>	316
6.85.3.9	<a href="#">show(tdt::uint, tdt::uint, bool=true)</a>	316
6.85.3.10	<a href="#">unlock(tdt::uint, tdt::uint)</a>	316
6.85.4	<a href="#">Member Data Documentation</a>	316
6.85.4.1	<a href="#">cols_</a>	316
6.85.4.2	<a href="#">prices_</a>	316
6.85.4.3	<a href="#">rows_</a>	317
6.85.4.4	<a href="#">script_</a>	317
6.85.4.5	<a href="#">unlocked_</a>	317
6.86	<a href="#">util::heuristic::RUN_AWAY_HEURISTIC Struct Reference</a>	317
6.86.1	<a href="#">Detailed Description</a>	318
6.87	<a href="#">lpp::Script Class Reference</a>	318
6.87.1	<a href="#">Detailed Description</a>	320
6.87.2	<a href="#">Constructor &amp; Destructor Documentation</a>	320
6.87.2.1	<a href="#">Script(const Script &amp;)=delete</a>	320
6.87.2.2	<a href="#">~Script()</a>	320
6.87.2.3	<a href="#">Script()</a>	320
6.87.3	<a href="#">Member Function Documentation</a>	320
6.87.3.1	<a href="#">call(const std::string &amp;fname, Args...as)</a>	320
6.87.3.2	<a href="#">call(const std::string &amp;fname)</a>	321
6.87.3.3	<a href="#">clear_stack()</a>	321
6.87.3.4	<a href="#">execute(const std::string &amp;)</a>	321
6.87.3.5	<a href="#">get(const std::string &amp;name)</a>	321
6.87.3.6	<a href="#">get_(const std::string &amp;name=""unknown"")</a>	321

6.87.3.7	<code>get_field_to_stack(const std::string &amp;)</code>	322
6.87.3.8	<code>get_stack_contents()</code>	323
6.87.3.9	<code>get_state()</code>	323
6.87.3.10	<code>get_vector(const std::string &amp;name)</code>	323
6.87.3.11	<code>instance()</code>	323
6.87.3.12	<code>is_nil(const std::string &amp;)</code>	323
6.87.3.13	<code>load(const std::string &amp;)</code>	324
6.87.3.14	<code>push_arg(Arg a)</code>	324
6.87.3.15	<code>push_arg(int arg)</code>	324
6.87.3.16	<code>push_args(Arg a, Args...as)</code>	324
6.87.3.17	<code>push_args(Arg a)</code>	324
6.87.3.18	<code>register_function(const std::string &amp;, lua_CFunction)</code>	325
6.87.3.19	<code>reload_all_scripts()</code>	325
6.87.3.20	<code>set(const std::string &amp;name, T val)</code>	325
6.87.4	Member Data Documentation	325
6.87.4.1	L	325
6.87.4.2	loaded_scripts_	326
6.88	SelectionBox Class Reference	326
6.88.1	Detailed Description	327
6.88.2	Constructor & Destructor Documentation	327
6.88.2.1	<code>SelectionBox(const Ogre::String &amp;, EntitySystem &amp;, Ogre::PlaneBounded↵ VolumeListSceneQuery &amp;, Ogre::RaySceneQuery &amp;, Ogre::SceneManager &amp;)</code>	327
6.88.2.2	<code>~SelectionBox()</code>	328
6.88.3	Member Function Documentation	328
6.88.3.1	<code>clear_selected_entities()</code>	328
6.88.3.2	<code>execute_selection(const Ogre::Vector2 &amp;, Ogre::Camera &amp;, bool=false)</code>	328
6.88.3.3	<code>execute_single_selection(Ogre::Camera &amp;)</code>	328
6.88.3.4	<code>extend_to(const Ogre::Vector2 &amp;)</code>	329
6.88.3.5	<code>get_selected_entities()</code>	329
6.88.3.6	<code>is_selecting() const</code>	329
6.88.3.7	<code>select_object(Ogre::MovableObject &amp;)</code>	329

6.88.3.8	<code>set_corners(tdt::real, tdt::real, tdt::real, tdt::real)</code>	329
6.88.3.9	<code>set_corners(const Ogre::Vector2 &amp;, const Ogre::Vector2 &amp;)</code>	330
6.88.3.10	<code>set_selecting(bool)</code>	330
6.88.3.11	<code>set_starting_point(const Ogre::Vector2 &amp;)</code>	330
6.88.4	Member Data Documentation	330
6.88.4.1	<code>entities_</code>	330
6.88.4.2	<code>ray_query_</code>	331
6.88.4.3	<code>scene_mgr_</code>	331
6.88.4.4	<code>selected_entities_</code>	331
6.88.4.5	<code>selection_in_progress_</code>	331
6.88.4.6	<code>start_</code>	331
6.88.4.7	<code>volume_query_</code>	331
6.89	Spellcaster::SPELL Struct Reference	331
6.89.1	Detailed Description	332
6.90	Spellcaster Class Reference	332
6.90.1	Detailed Description	333
6.90.2	Constructor & Destructor Documentation	333
6.90.2.1	<code>Spellcaster(EntityPlacer &amp;, SelectionBox &amp;)</code>	333
6.90.2.2	<code>~Spellcaster()</code> =default	333
6.90.3	Member Function Documentation	333
6.90.3.1	<code>cast(Ogre::Vector2=Ogre::Vector2{})</code>	333
6.90.3.2	<code>get_last_spell() const</code>	334
6.90.3.3	<code>get_last_spell_id() const</code>	334
6.90.3.4	<code>get_last_spell_type() const</code>	334
6.90.3.5	<code>get_spell() const</code>	334
6.90.3.6	<code>get_spell_type() const</code>	334
6.90.3.7	<code>is_casting() const</code>	334
6.90.3.8	<code>set_last_spell_id(tdt::uint)</code>	334
6.90.3.9	<code>set_spell(const std::string &amp;)</code>	335
6.90.3.10	<code>set_spell_type(SPELL_TYPE)</code>	335

6.90.3.11	<code>stop_casting()</code>	335
6.90.4	Member Data Documentation	335
6.90.4.1	<code>curr_spell_</code>	335
6.90.4.2	<code>last_spell_</code>	335
6.90.4.3	<code>last_spell_id_</code>	336
6.90.4.4	<code>placer_</code>	336
6.90.4.5	<code>script_</code>	336
6.90.4.6	<code>selector_</code>	336
6.91	SpellCastingWindow Class Reference	336
6.91.1	Detailed Description	338
6.91.2	Constructor & Destructor Documentation	338
6.91.2.1	<code>SpellCastingWindow()</code>	338
6.91.2.2	<code>~SpellCastingWindow()</code> =default	338
6.91.3	Member Function Documentation	338
6.91.3.1	<code>cast(int)</code>	338
6.91.3.2	<code>clear_spells()</code>	338
6.91.3.3	<code>deactivate_current_spell()</code>	338
6.91.3.4	<code>dec_selection()</code>	339
6.91.3.5	<code>get_spell(std::size_t)</code>	339
6.91.3.6	<code>get_spells() const</code>	339
6.91.3.7	<code>inc_selection()</code>	339
6.91.3.8	<code>init_()</code> override	339
6.91.3.9	<code>register_spell(const std::string &amp;)</code>	339
6.91.3.10	<code>set_caster(Spellcaster *)</code>	340
6.91.3.11	<code>set_spell_active(int)</code>	340
6.91.3.12	<code>update_selection_()</code>	340
6.91.4	Member Data Documentation	340
6.91.4.1	<code>caster_</code>	340
6.91.4.2	<code>curr_active_spell_</code>	340
6.91.4.3	<code>script_</code>	341



6.91.4.4	selection_number_ . . . . .	341
6.91.4.5	spells_ . . . . .	341
6.92	SpellComponent Struct Reference . . . . .	341
6.92.1	Detailed Description . . . . .	342
6.93	StructureComponent Struct Reference . . . . .	342
6.93.1	Detailed Description . . . . .	342
6.94	System Class Reference . . . . .	343
6.94.1	Detailed Description . . . . .	343
6.94.2	Constructor & Destructor Documentation . . . . .	344
6.94.2.1	~System()=default . . . . .	344
6.94.3	Member Function Documentation . . . . .	344
6.94.3.1	update(tdt::real)=0 . . . . .	344
6.95	TaskComponent Struct Reference . . . . .	344
6.95.1	Detailed Description . . . . .	345
6.96	TaskHandlerComponent Struct Reference . . . . .	345
6.96.1	Detailed Description . . . . .	346
6.97	TaskSystem Class Reference . . . . .	346
6.97.1	Detailed Description . . . . .	347
6.97.2	Constructor & Destructor Documentation . . . . .	347
6.97.2.1	TaskSystem(EntitySystem &, GridSystem &, CombatSystem &) . . . . .	347
6.97.2.2	~TaskSystem()=default . . . . .	347
6.97.3	Member Function Documentation . . . . .	347
6.97.3.1	current_task_completed_(tdt::uint, TaskHandlerComponent &) . . . . .	347
6.97.3.2	get_task_name(TASK_TYPE) const . . . . .	348
6.97.3.3	handle_task_(tdt::uint, TaskHandlerComponent &) . . . . .	348
6.97.3.4	next_task_(TaskHandlerComponent &) . . . . .	348
6.97.3.5	update(tdt::real) override . . . . .	348
6.97.4	Member Data Documentation . . . . .	349
6.97.4.1	combat_ . . . . .	349
6.97.4.2	entities_ . . . . .	349

6.97.4.3	grid_ . . . . .	349
6.97.4.4	task_names_ . . . . .	349
6.98	TimeComponent Struct Reference . . . . .	349
6.98.1	Detailed Description . . . . .	350
6.99	TimeSystem Class Reference . . . . .	350
6.99.1	Detailed Description . . . . .	351
6.99.2	Constructor & Destructor Documentation . . . . .	351
6.99.2.1	TimeSystem(EntitySystem &) . . . . .	351
6.99.2.2	~TimeSystem()=default . . . . .	351
6.99.3	Member Function Documentation . . . . .	351
6.99.3.1	advance_all_timers(tdt::real) . . . . .	351
6.99.3.2	advance_all_timers_of_type(tdt::real, TIME_EVENT) . . . . .	352
6.99.3.3	get_time_multiplier() . . . . .	352
6.99.3.4	handle_event_(tdt::uint, TimeComponent &) . . . . .	352
6.99.3.5	set_time_multiplier(tdt::real=1.f) . . . . .	352
6.99.3.6	update(tdt::real) override . . . . .	352
6.99.4	Member Data Documentation . . . . .	353
6.99.4.1	entities_ . . . . .	353
6.99.4.2	time_multiplier_ . . . . .	353
6.100	TopBar Class Reference . . . . .	353
6.100.1	Detailed Description . . . . .	354
6.100.2	Constructor & Destructor Documentation . . . . .	354
6.100.2.1	TopBar() . . . . .	354
6.100.2.2	~TopBar()=default . . . . .	354
6.100.3	Member Function Documentation . . . . .	354
6.100.3.1	init_() override . . . . .	354
6.100.3.2	update_label(const std::string &, const std::string &) . . . . .	354
6.100.3.3	update_time(tdt::real) . . . . .	355
6.100.4	Member Data Documentation . . . . .	355
6.100.4.1	tdelta_ . . . . .	355

6.101 TriggerComponent Struct Reference . . . . .	355
6.101.1 Detailed Description . . . . .	356
6.102 TriggerSystem Class Reference . . . . .	356
6.102.1 Detailed Description . . . . .	357
6.102.2 Constructor & Destructor Documentation . . . . .	357
6.102.2.1 TriggerSystem(EntitySystem &) . . . . .	357
6.102.2.2 ~TriggerSystem()=default . . . . .	357
6.102.3 Member Function Documentation . . . . .	357
6.102.3.1 get_check_period() const . . . . .	357
6.102.3.2 set_check_period(tdt::real) . . . . .	357
6.102.3.3 update(tdt::real) override . . . . .	358
6.102.4 Member Data Documentation . . . . .	358
6.102.4.1 check_timer_ . . . . .	358
6.102.4.2 entities_ . . . . .	358
6.103 UpgradeComponent Struct Reference . . . . .	358
6.103.1 Detailed Description . . . . .	359
6.104 WaveSystem Class Reference . . . . .	359
6.104.1 Detailed Description . . . . .	362
6.104.2 Constructor & Destructor Documentation . . . . .	362
6.104.2.1 WaveSystem(EntitySystem &) . . . . .	362
6.104.2.2 ~WaveSystem()=default . . . . .	363
6.104.3 Member Function Documentation . . . . .	363
6.104.3.1 add_entity_blueprint(const std::string &) . . . . .	363
6.104.3.2 add_spawn_node(tdt::uint) . . . . .	363
6.104.3.3 advance_wave_countdown(tdt::uint) . . . . .	363
6.104.3.4 clear_entity_blueprints() . . . . .	363
6.104.3.5 clear_spawn_nodes() . . . . .	364
6.104.3.6 end_wave() . . . . .	364
6.104.3.7 get_countdown_value() const . . . . .	364
6.104.3.8 get_curr_wave_number() const . . . . .	364

6.104.3.9	<code>get_endless_mode()</code> const	364
6.104.3.10	<code>get_entities_spawned()</code> const	364
6.104.3.11	<code>get_entity_blueprints()</code> const	364
6.104.3.12	<code>get_entity_total()</code> const	364
6.104.3.13	<code>get_spawn_cooldown()</code> const	365
6.104.3.14	<code>get_spawn_timer()</code> const	365
6.104.3.15	<code>get_spawning_nodes()</code> const	365
6.104.3.16	<code>get_state()</code> const	365
6.104.3.17	<code>get_wave_count()</code> const	365
6.104.3.18	<code>get_wave_entities()</code> const	365
6.104.3.19	<code>get_wave_table()</code> const	365
6.104.3.20	<code>next_wave()</code>	365
6.104.3.21	<code>pause(bool)</code>	365
6.104.3.22	<code>set_countdown_value(tdt::uint)</code>	366
6.104.3.23	<code>set_countdown_window(CEGUI::Window *)</code>	366
6.104.3.24	<code>set_curr_wave_number(tdt::uint)</code>	366
6.104.3.25	<code>set_endless_mode(bool)</code>	366
6.104.3.26	<code>set_entities_spawned(tdt::uint)</code>	367
6.104.3.27	<code>set_entity_total(tdt::uint)</code>	367
6.104.3.28	<code>set_spawn_cooldown(tdt::real)</code>	367
6.104.3.29	<code>set_spawn_timer(tdt::real)</code>	367
6.104.3.30	<code>set_state(WAVE_STATE)</code>	368
6.104.3.31	<code>set_wave_count(tdt::uint)</code>	368
6.104.3.32	<code>set_wave_entities(tdt::uint)</code>	368
6.104.3.33	<code>set_wave_table(const std::string &amp;)</code>	368
6.104.3.34	<code>spawn()</code>	368
6.104.3.35	<code>start()</code>	369
6.104.3.36	<code>start_wave()</code>	369
6.104.3.37	<code>update(tdt::real) override</code>	369
6.104.3.38	<code>update_label_text()</code>	369

6.104.3.39	wave_entity_died()	369
6.104.4	Member Data Documentation	369
6.104.4.1	curr_wave_number_	369
6.104.4.2	endless_mode_	370
6.104.4.3	entities_	370
6.104.4.4	entities_spawned_	370
6.104.4.5	entities_total_	370
6.104.4.6	entity_blueprints_	370
6.104.4.7	label_text_	370
6.104.4.8	next_wave_countdown_	370
6.104.4.9	script_	370
6.104.4.10	second_timer_	371
6.104.4.11	spawn_timer_	371
6.104.4.12	spawning_nodes_	371
6.104.4.13	state_	371
6.104.4.14	wave_count_	371
6.104.4.15	wave_entities_	371
6.104.4.16	wave_table_	371
6.104.4.17	window_	371
<b>Index</b>		<b>373</b>



# Chapter 1

## The Dungeon Throne

Dzejrou, MFF UK

### Thesis Name:

The Dungeon Throne: A 3D Dungeon Managment [Game](#)

### Abstract:

The goal of this thesis is to design and implement a real-time strategy game in a 3D world with emphasis on high modifiability using a suitable scripting language. Inspired by the Dungeon Keeper series developed by Bullfrog Studios, the player's goal in this game is to protect his dungeon from endless armies of heroes raiding his domain with intentions to steal his treasures.

Once finished, the game's scripting engine should offer the ability to change data and logic of entities and systems to people with at least a basic understanding of programming. This will lead to easy future extensibility of the game and the possibility to create easily installable modifications.

### Literature:

- Programming In Lua, 3rd Edition, Roberto Ierusalimsky, Lua.org 2013
- [Game](#) Engine Architecture, Jason Gregory, A K Peters/CRC Press 2014
- Programming [Game](#) AI By Example, Mat Buckland, Wordware Publishing Inc. 2005

### Progress album:

<http://imgur.com/a/PDMd7>





## Chapter 2

# Namespace Index

### 2.1 Namespace List

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

<a href="#">action</a>	Functions representing actions that can be key bound . . . . .	17
<a href="#">AIHelper</a>	Namespace containing auxiliary functions that help with the management of the ai component . . . . .	19
<a href="#">CombatHelper</a>	Namespace containing auxiliary functions that help with the management of the combat component . . . . .	20
<a href="#">CommandHelper</a>	Namespace containing auxiliary functions that help with the management of the command component . . . . .	25
<a href="#">ConstructorHelper</a>	Namespace containing auxiliary functions that help with constructor component management . . . . .	27
<a href="#">CounterHelper</a>	Namespace containing auxiliary functions that help with counter component management . . . . .	28
<a href="#">DestructorHelper</a>	Namespace containing auxiliary functions that help with the management of the destructor component . . . . .	30
<a href="#">EventHandlerHelper</a>	Namespace containing auxiliary functions that help with the management of the event handler component . . . . .	32
<a href="#">EventHelper</a>	Auxiliary namespace containing functions that help with the management of the event component . . . . .	34
<a href="#">ExperienceValueHelper</a>	Namespace containing auxiliary functions that help with the management of the experience value component . . . . .	37
<a href="#">ExplosionHelper</a>	Namespace containing auxiliary functions that help with the management of the explosion component . . . . .	38
<a href="#">FactionHelper</a>	Auxiliary namespace containing functions that help with the management of the faction component . . . . .	41
<a href="#">GoldHelper</a>	Auxiliary namespace containing functions that help with the management of the gold component . . . . .	42
<a href="#">GraphicsHelper</a>	Auxiliary namespace containing functions that help with the management of the graphics component . . . . .	45

<a href="#">GridNodeHelper</a>	Auxiliary namespace containing functions that help with the management of the grid node component . . . . .	51
<a href="#">HealthHelper</a>	Auxiliary namespace containing functions that help with the management of the health component . . . . .	54
<a href="#">HomingHelper</a>	Auxiliary namespace containing functions that help with the management of the homing component . . . . .	60
<a href="#">InputHelper</a>	Auxiliary namespace containing functions that help with the management of the input component . . . . .	62
<a href="#">level_generators</a>	Namespace that contains different level generators that can be used in the game as well as the typedef for the DEFAULT_LEVEL_GENERATOR . . . . .	63
<a href="#">LightHelper</a>	Namespace containing auxiliary functions that help with the management of the light component . . . . .	63
<a href="#">LimitedLifeSpanHelper</a>	Auxiliary namespace containing functions that help with the management of the limited life span component . . . . .	65
<a href="#">ManaCrystalHelper</a>	Auxiliary namespace that contains functions that help with the management of the mana crystal component . . . . .	66
<a href="#">ManaHelper</a>	Auxiliary namespace that contains functions that help with the management of the mana component . . . . .	68
<a href="#">MovementHelper</a>	Auxiliary namespace containing functions that help with the management of the movement component . . . . .	71
<a href="#">NameHelper</a>	Auxiliary namespace containing functions that help with the management of the name component . . . . .	74
<a href="#">NotificationHelper</a>	Auxiliary namespace containing functions that help with the management of the notification component . . . . .	75
<a href="#">OnHitHelper</a>	Auxiliary namespace containing functions that help with the management of the on hit component . . . . .	77
<a href="#">PathfindingHelper</a>	Auxiliary namespace containing functions that help with the management of the pathfinding component . . . . .	79
<a href="#">PhysicsHelper</a>	Auxiliary namespace containing functions that help with the management of the physics component . . . . .	81
<a href="#">PriceHelper</a>	Auxiliary namespace containing functions that help with the management of the price component . . . . .	85
<a href="#">ProductHelper</a>	Auxiliary namespace containing functions that help with the management of the product component . . . . .	85
<a href="#">ProductionHelper</a>	Auxiliary namespace containing functions that help with the management of the production component . . . . .	86
<a href="#">SpellHelper</a>	Auxiliary namespace that contains functions that help with the management of the spell component . . . . .	90
<a href="#">StructureHelper</a>	Auxiliary namespace containing the functions that help with the management of the structure component . . . . .	93

<a href="#">TaskHandlerHelper</a>	Auxiliary namespace containing functions that help with the management of the task handler component . . . . .	95
<a href="#">TaskHelper</a>	Auxiliary namespace containing functions that help with the management of the task component . . . . .	97
<a href="#">tdt</a>	Namespace containing numeric types used in the game . . . . .	101
<a href="#">TimeHelper</a>	Auxiliary namespace containing functions that help with the management of the time component . . . . .	101
<a href="#">TriggerHelper</a>	Auxiliary namespace containing functions that help with the management of the trigger component . . . . .	104
<a href="#">UpgradeHelper</a>	Auxiliary namespace containing functions that help with the management of the upgrade component . . . . .	108
<a href="#">util</a>	The util namespace contains functors used as conditions in searches and other helper structures/functions used throughout the code . . . . .	112
<a href="#">util::effect</a>	Contains effect functors that perform an action on the entity they are called on . . . . .	115
<a href="#">util::heuristic</a>	Forward declarations . . . . .	115
<a href="#">util::path_type</a>	Contains different path types, which are used to check if a path should be returned once found or when an augmenting edge is found to the path . . . . .	116
<a href="#">util::pathfinding</a>	Contains the pathfinding algorithms used by the <a href="#">util::pathfind</a> function . . . . .	116



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

util::pathfinding::A_STAR< PATH_TYPE > . . . . .	117
AIComponent . . . . .	118
AlignComponent . . . . .	121
AlignComponent::AlignState . . . . .	122
util::path_type::BEST_PATH . . . . .	122
Camera . . . . .	126
CombatComponent . . . . .	133
CommandComponent . . . . .	144
Component . . . . .	145
ConstructorComponent . . . . .	149
CounterComponent . . . . .	150
util::effect::DAMAGE_EFFECT . . . . .	151
DestructorComponent . . . . .	152
util::EntityDestroyer . . . . .	156
EntityPlacer . . . . .	157
EventComponent . . . . .	181
EventHandlerComponent . . . . .	182
lpp::Exception . . . . .	186
ExperienceValueComponent . . . . .	187
ExplosionComponent . . . . .	188
FactionComponent . . . . .	189
util::path_type::FIRST_PATH . . . . .	190
FrameListener	
Game . . . . .	192
util::effect::FREEZE_EFFECT . . . . .	190
GameSerializer . . . . .	204
GoldComponent . . . . .	209
GraphicsComponent . . . . .	210
Grid . . . . .	213
GridNodeComponent . . . . .	221
GUI . . . . .	225
GUIWindow . . . . .	234
BuilderWindow . . . . .	122
Console . . . . .	145
EntityCreator . . . . .	153

EntityTracker . . . . .	177
GameLog . . . . .	201
MessageToPlayerWindow . . . . .	276
OptionsWindow . . . . .	286
ResearchWindow . . . . .	312
SpellCastingWindow . . . . .	336
TopBar . . . . .	353
util::HAS_GOLD . . . . .	237
util::effect::HEAL_EFFECT . . . . .	238
HealthComponent . . . . .	240
util::heuristic::HEURISTIC . . . . .	243
util::heuristic::MANHATTAN_DISTANCE . . . . .	275
util::heuristic::PORTAL_HEURISTIC . . . . .	301
util::heuristic::NO_HEURISTIC . . . . .	284
util::heuristic::RUN_AWAY_HEURISTIC . . . . .	317
HomingComponent . . . . .	244
InputComponent . . . . .	245
util::IS_ENEMY . . . . .	250
util::IS_FRIENDLY . . . . .	251
util::IS_FRIENDLY_OR_NEUTRAL . . . . .	253
util::IS_GOLD_VAULT . . . . .	255
KeyListener . . . . .	
Game . . . . .	192
level_generators::LevelGenerator . . . . .	256
level_generators::RandomLevelGenerator . . . . .	309
LightComponent . . . . .	258
LimitedLifeSpanComponent . . . . .	259
util::effect::LOWER_SPEED_EFFECT . . . . .	259
LuaInterface . . . . .	261
ManaComponent . . . . .	272
ManaCrystalComponent . . . . .	272
ManualObject . . . . .	
SelectionBox . . . . .	326
MineComponent . . . . .	279
MouseListener . . . . .	
Game . . . . .	192
MovementComponent . . . . .	279
NameComponent . . . . .	283
NotificationComponent . . . . .	284
OnHitComponent . . . . .	285
PathfindingComponent . . . . .	291
PhysicsComponent . . . . .	291
Player . . . . .	292
PortalComponent . . . . .	302
PriceComponent . . . . .	302
ProductComponent . . . . .	303
ProductionComponent . . . . .	304
util::path_type::RANDOM_PATH< UPPER > . . . . .	308
RayCaster . . . . .	310
lpp::Script . . . . .	318
Spellcaster::SPELL . . . . .	331
Spellcaster . . . . .	332
SpellComponent . . . . .	341
StructureComponent . . . . .	342
System . . . . .	343
AISystem . . . . .	118
CombatSystem . . . . .	134

EntitySystem . . . . .	162
EventSystem . . . . .	183
GraphicsSystem . . . . .	211
GridSystem . . . . .	222
HealthSystem . . . . .	240
InputSystem . . . . .	246
ManaSpellSystem . . . . .	273
MovementSystem . . . . .	280
ProductionSystem . . . . .	305
TaskSystem . . . . .	346
TimeSystem . . . . .	350
TriggerSystem . . . . .	356
WaveSystem . . . . .	359
TaskComponent . . . . .	344
TaskHandlerComponent . . . . .	345
TimeComponent . . . . .	349
TriggerComponent . . . . .	355
UpgradeComponent . . . . .	358
WindowEventListener	
Game . . . . .	192





## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">util::pathfinding::A_STAR&lt; PATH_TYPE &gt;</a>	Simple A* pathfinding implementations with path type specified as a template parameter . . .	117
<a href="#">AIComponent</a>	Holds info about the base Lua table to be called for initializing, updating and finishing an entity as well as couple categorizing enums . . . . .	118
<a href="#">AISystem</a>	<a href="#">System</a> handling the AI of entities by calling their update method every frame . . . . .	118
<a href="#">AlignComponent</a>	Holds information about an objects align states, i.e . . . . .	121
<a href="#">AlignComponent::AlignState</a>	. . . . .	122
<a href="#">util::path_type::BEST_PATH</a>	Finds the best path by refusing any paths found . . . . .	122
<a href="#">BuilderWindow</a>	Class representing the building selection window, allows the player to place registered (unlocked) buildings . . . . .	122
<a href="#">Camera</a>	Class wrapping the Ogre camera object, allowing RTS-like movement and switching to free mode	126
<a href="#">CombatComponent</a>	Holds info about an entity's attack types and damage . . . . .	133
<a href="#">CombatSystem</a>	Manages auto attack melee and ranged combat, special melee and ranged attacks will be both handled by the spellcasting system . . . . .	134
<a href="#">CommandComponent</a>	Contains a list of commands an entity can respond to . . . . .	144
<a href="#">Component</a>	. . . . .	145
<a href="#">Console</a>	Class representing the ingame developers console that allows for runtime execution of Lua code	145
<a href="#">ConstructorComponent</a>	Contains the blueprint that gets called when an entity that has this component is created . . .	149
<a href="#">CounterComponent</a>	A simple incrementing counter . . . . .	150
<a href="#">util::effect::DAMAGE_EFFECT</a>	Deals a random damage in a given range to the entity it's called on . . . . .	151
<a href="#">DestructorComponent</a>	Contains name of the table that contains the function (called "dtor") which is called when an entity is destroyed . . . . .	152

<a href="#">EntityCreator</a>	Class representing the debugging <a href="#">GUI</a> window used to place and create entities during runtime	153
<a href="#">util::EntityDestroyer</a>	A structure providing the private method <a href="#">EntitySystem::destroy_entity</a> to the <a href="#">DestructorHelper</a> ↔ ::destroy function	156
<a href="#">EntityPlacer</a>	Class allowing the player/developer to place entities on the ground	157
<a href="#">EntitySystem</a>	Handles everything related to entities, like addition and removal of components, testing if an entity has a component or retrieval of components belonging to particular entities	162
<a href="#">EntityTracker</a>	A window that monitors the stats of the currently selected entity and allows for it's upgrading once it has enough experience	177
<a href="#">EventComponent</a>	Represents events that happen in the game, like gold seams dropping gold, curing an entity of poisoning or triggers from traps etc	181
<a href="#">EventHandlerComponent</a>	Allows to cherry pick when it comes to event handling and handle only certain events	182
<a href="#">EventSystem</a>		183
<a href="#">lpp::Exception</a>	<a href="#">Exception</a> class used to throw exception from the <a href="#">Script</a> class	186
<a href="#">ExperienceValueComponent</a>	The amount of experience the entity yields when killed	187
<a href="#">ExplosionComponent</a>	<a href="#">Component</a> used to create the visual effect of an explosion, the damage should be done in the explosion's constructor so that it's not applied on each frame	188
<a href="#">FactionComponent</a>	Represents the faction an entity that has this component if a member of	189
<a href="#">util::path_type::FIRST_PATH</a>	Finds the first path by accepting the first path found	190
<a href="#">util::effect::FREEZE_EFFECT</a>	Freezes a given entity in place for a given time period	190
<a href="#">Game</a>		192
<a href="#">GameLog</a>	Class representing the log window used to show messages to the player	201
<a href="#">GameSerializer</a>	Class that is used to save (by using Lua code generation) and loading the game (by executing said code)	204
<a href="#">GoldComponent</a>	Represents a gold amount an entity is holding, be it a gold seam, worker minion or gold depository	209
<a href="#">GraphicsComponent</a>	Holds info related to the Ogre3D rendering library	210
<a href="#">GraphicsSystem</a>	<a href="#">System</a> that performs all graphics related updates	211
<a href="#">Grid</a>	Class representing the pathfinding grid	213
<a href="#">GridNodeComponent</a>	Holds GridNode's neighbour nodes	221
<a href="#">GridSystem</a>	Represents the pathfinding graph used by the game and provides several methods related to pathfinding that can be used in Lua	222
<a href="#">GUI</a>	Represents the game's main graphical user interface (i.e	225
<a href="#">GUIWindow</a>	Abstract class that custom <a href="#">GUI</a> windows inherit from, prevents unnecessary rewriting of common functions (like visibility setting and window_ assignment on init)	234

<a href="#">util::HAS_GOLD</a>	
Tests if a given entity has a gold component . . . . .	237
<a href="#">util::effect::HEAL_EFFECT</a>	
Fully heals the entity it's called on . . . . .	238
<a href="#">HealthComponent</a>	
Holds info about an entity's health and regeneration . . . . .	240
<a href="#">HealthSystem</a>	
<a href="#">System</a> that manages the regeneration and health of entities on each frame . . . . .	240
<a href="#">util::heuristic::HEURISTIC</a>	
Abstract parent of all heuristics . . . . .	243
<a href="#">HomingComponent</a>	
Used for projectiles that are supposed to follow a target and deal damage when they hit it . . .	244
<a href="#">InputComponent</a>	
Holds info related to direct player input applied to an entity . . . . .	245
<a href="#">InputSystem</a>	
<a href="#">System</a> handling entities controlled by the player's keyboard input and changing the game's view mode (between 1st and 3rd person) . . . . .	246
<a href="#">util::IS_ENEMY</a>	
Tests if if the entity it is called on is an enemy of the entity specified in it's constructor . . . . .	250
<a href="#">util::IS_FRIENDLY</a>	
Tests if if the entity it is called on is a friend of the entity specified in it's constructor . . . . .	251
<a href="#">util::IS_FRIENDLY_OR_NEUTRAL</a>	
Tests if if the entity it is called on is a friend of or neutral to the entity specified in it's constructor	253
<a href="#">util::IS_GOLD_VAULT</a>	
Tests if a given entity is of friendly faction, has structure component and has gold component (that is, it's a gold vault) . . . . .	255
<a href="#">level_generators::LevelGenerator</a>	
Abstract parent class of all level generators, allows for different level generators used to create levels with minimal effort . . . . .	256
<a href="#">LightComponent</a>	
Allows an entity to emit light to it's surrounding area . . . . .	258
<a href="#">LimitedLifeSpanComponent</a>	
Allows to create entities that are automatically killed (summons) after a certain amount of time has passed (lifespan) . . . . .	259
<a href="#">util::effect::LOWER_SPEED_EFFECT</a>	
Halves the speed of the entity it's called on for a given time period . . . . .	259
<a href="#">LuaInterface</a>	
Class that creates an interface between engine (C++) and logic (Lua) code . . . . .	261
<a href="#">ManaComponent</a>	
Allows an entity to cast spell by providing the mana resource . . . . .	272
<a href="#">ManaCrystalComponent</a>	
Allows an entity to increase the player's mana capacity and regeneration rate while it's alive . .	272
<a href="#">ManaSpellSystem</a>	
Regenerates mana to the player and all entities that have mana . . . . .	273
<a href="#">util::heuristic::MANHATTAN_DISTANCE</a>	
Returns the manhattan distance between two nodes . . . . .	275
<a href="#">MessageToPlayerWindow</a>	
A window that can show the player a text message with 1, 2 or 3 buttons (custom labels) that can call assigned functions . . . . .	276
<a href="#">MineComponent</a>	
Dummy component that signals that an entity having it can be mined . . . . .	279
<a href="#">MovementComponent</a>	
Holds info related to movement, if an entity has this component it should also have a Physics component (containing the entity's position), otherwise the <a href="#">MovementSystem</a> might not work correctly . . . . .	279
<a href="#">MovementSystem</a>	
<a href="#">System</a> handling movement related updates and containing movement & physics related methods . . . . .	280

<a href="#">NameComponent</a>	
Name of the entity shown in the entity viewer . . . . .	283
<a href="#">util::heuristic::NO_HEURISTIC</a>	
Represents no heuristic by returning 0 all the time . . . . .	284
<a href="#">NotificationComponent</a>	
Allows to keep track about notification cooldown, so that an entity doesn't spam the player with messages on reoccurring events in a short time period . . . . .	284
<a href="#">OnHitComponent</a>	
Contains the blueprint table which gets called when an entity that has this component gets hit . . . . .	285
<a href="#">OptionsWindow</a>	
Options menu window that lets the player to change the resolution, window mode and keybinding . . . . .	286
<a href="#">PathfindingComponent</a>	
Holds data related to the entity's current path . . . . .	291
<a href="#">PhysicsComponent</a>	
Components . . . . .	291
<a href="#">Player</a>	
Auxiliary class representing the player's resources, since the nature of the game does not allow creating player as an entity (only one player can exist at a time) and it allows for easy <a href="#">G↔UI</a> modifications when the amount of these resources changes (had player been an entity, ever gold/mana addition would require a check which would be true only in minimal number of cases) . . . . .	292
<a href="#">util::heuristic::PORTAL_HEURISTIC</a>	
Variation of the Manhattan distance heuristic that takes portals into accounts . . . . .	301
<a href="#">PortalComponent</a>	
Dummy component that signals that an entity having it is a portal - which is used in pathfinding . . . . .	302
<a href="#">PriceComponent</a>	
Represents either gold or mana cost of an entity . . . . .	302
<a href="#">ProductComponent</a>	
References the producer of the entity that has this component . . . . .	303
<a href="#">ProductionComponent</a>	
Allows scheduled production of new entities (spawners) of a given type up to a maximum amount . . . . .	304
<a href="#">ProductionSystem</a>	
System taking care of entities spawned by buildings and the spawn counts allowing for a constant amount of entities (related to the number of buildings spawning entities of that blueprint table) . . . . .	305
<a href="#">util::path_type::RANDOM_PATH&lt; UPPER &gt;</a>	
Finds a random path by returning true only when a random number in the range (0, UPPER) is equal to 0 . . . . .	308
<a href="#">level_generators::RandomLevelGenerator</a>	
Level generator that uses simple RNG approach (counts the number of gold neighbours and increases the chance to spawn a gold deposit if needed) . . . . .	309
<a href="#">RayCaster</a>	
Manages polygon precise raycasting used with half walls that have empty spaces in their bounding boxes . . . . .	310
<a href="#">ResearchWindow</a>	
Class that represents the research window in the game, which allows the player to unlock new buildings and spells . . . . .	312
<a href="#">util::heuristic::RUN_AWAY_HEURISTIC</a>	
Used by entities that want to run away from an enemy . . . . .	317
<a href="#">lpp::Script</a>	
Class representing a Lua script, allows to register C++ functions, load variables, call functions, execute strings containing Lua code and other functionalities . . . . .	318
<a href="#">SelectionBox</a>	
Class representing the ingame selection box (created by moving the left mouse while pressing the left mouse button) to select multiple entities on screen or a single entity (by simply clicking) . . . . .	326
<a href="#">Spellcaster::SPELL</a>	
A structure representing a spell by containing it's type and name . . . . .	331
<a href="#">Spellcaster</a>	
A utility class that manages the player's spell casting and is usually called from input handlers and the spell casting window . . . . .	332

<a href="#">SpellCastingWindow</a>	
Class representing the spell selection window, allows the player to cast registered (unlocked) spells . . . . .	336
<a href="#">SpellComponent</a>	
Allows an entity to periodically cast a spell . . . . .	341
<a href="#">StructureComponent</a>	
Defines a building (or a wall), by holding it's radius (of the area it takes in the grid) and vector of nodes that it sits on . . . . .	342
<a href="#">System</a>	
Parent class of all systems . . . . .	343
<a href="#">TaskComponent</a>	
Defines a task by giving it a type, source (the task handler) and a target (subject of the task) . . . . .	344
<a href="#">TaskHandlerComponent</a>	
Task queue and register of possible tasks, every entity that is able to actually do something on it's own should have it . . . . .	345
<a href="#">TaskSystem</a>	
<a href="#">System</a> managing all entities with the <a href="#">TaskComponent</a> , their creation, assignment, lifetime checks and canceling . . . . .	346
<a href="#">TimeComponent</a>	
Represents a timer that after a certain amount of time can start end an event (it's target) . . . . .	349
<a href="#">TimeSystem</a> . . . . .	350
<a href="#">TopBar</a>	
Class representing an info bar on the top of the screen displaying the name of the game, player's gold, mana, units and the current time . . . . .	353
<a href="#">TriggerComponent</a>	
Allows an entity to cause an effect (by calling it's blueprint) when its triggered (stepped on) or can notify a linked entity which causes the effect . . . . .	355
<a href="#">TriggerSystem</a>	
Handles triggers by checking if an entity is standing in their radius when they are off cooldowns . . . . .	356
<a href="#">UpgradeComponent</a>	
Represents the game's leveling system component, contains info about experience and leveling progression as well as the blueprint that gets called on level up . . . . .	358
<a href="#">WaveSystem</a>	
This system creates the entities attacking the player's dungeon in a similar fashion to tower defense games . . . . .	359



## Chapter 5

# Namespace Documentation

### 5.1 action Namespace Reference

Functions representing actions that can be key bound.

#### Functions

- void [CAST\\_SPELL\\_1](#) ()  
*Casts the first spell in the spell tool window.*
- void [CAST\\_SPELL\\_2](#) ()  
*Casts the second spell in the spell tool window.*
- void [CAST\\_SPELL\\_3](#) ()  
*Casts the third spell in the spell tool window.*
- void [CAST\\_SPELL\\_4](#) ()  
*Casts the fourth spell in the spell tool window.*
- void [NEXT](#) ()  
*Moves the spell selection to the right.*
- void [PREV](#) ()  
*Moves the spell selection to the left.*
- void [SPELL\\_TAB](#) ()  
*Switches the current tool to the spell tab.*
- void [BUILD\\_TAB](#) ()  
*Switches the current tool to the build tab.*
- void [MENU\\_TAB](#) ()  
*Switches the current tool to the menu tab.*
- void [RESET\\_CAMERA](#) ()  
*Resets the position and orientation of the camera.*
- void [QUICK\\_SAVE](#) ()  
*Saves the current game to the quick\_save.lua file.*
- void [QUICK\\_LOAD](#) ()  
*Restores the game save in the quick\_save.lua file.*

#### 5.1.1 Detailed Description

Functions representing actions that can be key bound.

## 5.1.2 Function Documentation

### 5.1.2.1 void action::BUILD\_TAB ( )

Switches the current tool to the build tab.

Definition at line 493 of file OptionsWindow.cpp.

### 5.1.2.2 void action::CAST\_SPELL\_1 ( )

Casts the first spell in the spell tool window.

Definition at line 438 of file OptionsWindow.cpp.

### 5.1.2.3 void action::CAST\_SPELL\_2 ( )

Casts the second spell in the spell tool window.

Definition at line 446 of file OptionsWindow.cpp.

### 5.1.2.4 void action::CAST\_SPELL\_3 ( )

Casts the third spell in the spell tool window.

Definition at line 454 of file OptionsWindow.cpp.

### 5.1.2.5 void action::CAST\_SPELL\_4 ( )

Casts the fourth spell in the spell tool window.

Definition at line 462 of file OptionsWindow.cpp.

### 5.1.2.6 void action::MENU\_TAB ( )

Switches the current tool to the menu tab.

Definition at line 500 of file OptionsWindow.cpp.

### 5.1.2.7 void action::NEXT ( )

Moves the spell selection to the right.

Definition at line 470 of file OptionsWindow.cpp.



**5.1.2.8 void action::PREV ( )**

Moves the spell selection to the left.

Definition at line 478 of file OptionsWindow.cpp.

**5.1.2.9 void action::QUICK\_LOAD ( )**

Restores the game save in the quick\_save.lua file.

Definition at line 521 of file OptionsWindow.cpp.

**5.1.2.10 void action::QUICK\_SAVE ( )**

Saves the current game to the quick\_save.lua file.

Definition at line 514 of file OptionsWindow.cpp.

**5.1.2.11 void action::RESET\_CAMERA ( )**

Resets the position and orientation of the camera.

Definition at line 507 of file OptionsWindow.cpp.

**5.1.2.12 void action::SPELL\_TAB ( )**

Switches the current tool to the spell tab.

Definition at line 486 of file OptionsWindow.cpp.

## 5.2 AIHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the ai component.

### Functions

- void [set\\_blueprint](#) ([EntitySystem](#) &, tdt::uint, const std::string &)  
*Changes the blueprint table name of a given entity.*
- const std::string & [get\\_blueprint](#) ([EntitySystem](#) &, tdt::uint)  
*Returns name of the blueprint table of a given entity (i.e.*
- void [set\\_state](#) ([EntitySystem](#) &, tdt::uint, ENTITY\_STATE::VAL)  
*Changes the state of a given entity.*
- ENTITY\_STATE::VAL [get\\_state](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the state a given entity is in.*

### 5.2.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the ai component.

### 5.2.2 Function Documentation

**5.2.2.1 const std::string & AIHelper::get\_blueprint ( [EntitySystem](#) & *ents*, tdt::uint *id* )**

Returns name of the blueprint table of a given entity (i.e.

the table containing it's init, update and finish methods).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 12 of file AIHelper.cpp.

#### 5.2.2.2 ENTITY\_STATE::VAL AIHelper::get\_state ( EntitySystem & ents, tdt::uint id )

Returns the state a given entity is in.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 30 of file AIHelper.cpp.

#### 5.2.2.3 void AIHelper::set\_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )

Changes the blueprint table name of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Name</i>	of the new blueprint table.

Definition at line 5 of file AIHelper.cpp.

#### 5.2.2.4 void AIHelper::set\_state ( EntitySystem & ents, tdt::uint id, ENTITY\_STATE::VAL val )

Changes the state of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>New</i>	state.

Definition at line 23 of file AIHelper.cpp.

## 5.3 CombatHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the combat component.

## Functions

- void [set\\_target](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Changes the target of a given entity's attack.*
- tdt::uint [get\\_target](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the target of a given entity's attack.*
- void [set\\_range](#) ([EntitySystem](#) &, tdt::uint, tdt::real)  
*Changes the attack range of a given entity.*
- tdt::real [get\\_range](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the attack range of a given entity.*
- void [set\\_dmg\\_range](#) ([EntitySystem](#) &, tdt::uint, tdt::uint, tdt::uint)  
*Changes the damage range (min damage, max damage) that a given entity can deal when attacking.*
- std::tuple< tdt::uint, tdt::uint > [get\\_dmg\\_range](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the damage range (in the form of a 2-member tuple) of a given entity.*
- tdt::uint [get\\_dmg](#) (tdt::uint, tdt::uint)  
*Returns a pseudo random damage value between given two numbers, used to calculate the damage of each individual attack.*
- void [set\\_cooldown](#) ([EntitySystem](#) &, tdt::uint, tdt::real)  
*Changes the cooldown (minimal time between attacks) of a given entity.*
- tdt::real [get\\_cooldown](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the cooldown (minimal time between attacks) of a given entity.*
- void [set\\_atk\\_type](#) ([EntitySystem](#) &, tdt::uint, ATTACK\_TYPE)  
*Changes the attack type of a given entity.*
- ATTACK\_TYPE [get\\_atk\\_type](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the attack type of a given entity.*
- bool [in\\_range](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Returns true if a given entity is in attack range from another entity.*
- void [set\\_projectile\\_blueprint](#) ([EntitySystem](#) &, tdt::uint, const std::string &)  
*Sets the projectile table used when a given entity shoots.*
- const std::string & [get\\_projectile\\_blueprint](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the projectile table used when a given entity shoots.*

### 5.3.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the combat component.

### 5.3.2 Function Documentation

#### 5.3.2.1 ATTACK\_TYPE CombatHelper::get\_atk\_type ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the attack type of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 87 of file `CombatHelper.cpp`.

### 5.3.2.2 tdt::real CombatHelper::get\_cooldown ( EntitySystem & ents, tdt::uint id )

Returns the cooldown (minimal time between attacks) of a given entity.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 71 of file CombatHelper.cpp.

### 5.3.2.3 tdt::uint CombatHelper::get\_dmg ( tdt::uint min, tdt::uint max )

Returns a pseudo random damage value between given two numbers, used to calculate the damage of each individual attack.

#### Parameters

<i>Minimal</i>	damage value.
<i>Maximal</i>	damage value.

Definition at line 59 of file CombatHelper.cpp.

### 5.3.2.4 std::tuple< tdt::uint, tdt::uint > CombatHelper::get\_dmg\_range ( EntitySystem & ents, tdt::uint id )

Returns the damage range (in the form of a 2-member tuple) of a given entity.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 50 of file CombatHelper.cpp.

### 5.3.2.5 const std::string & CombatHelper::get\_projectile\_blueprint ( EntitySystem & ents, tdt::uint id )

Returns the projectile table used when a given entity shoots.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 115 of file CombatHelper.cpp.

#### 5.3.2.6 tdt::real CombatHelper::get\_range ( EntitySystem & ents, tdt::uint id )

Returns the attack range of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 31 of file CombatHelper.cpp.

#### 5.3.2.7 tdt::uint CombatHelper::get\_target ( EntitySystem & ents, tdt::uint id )

Returns the target of a given entity's attack.

##### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 15 of file CombatHelper.cpp.

#### 5.3.2.8 bool CombatHelper::in\_range ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )

Returns true if a given entity is in attack range from another entity.

##### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity checking range.
<i>ID</i>	of the second entity.

Definition at line 96 of file CombatHelper.cpp.

#### 5.3.2.9 void CombatHelper::set\_atk\_type ( EntitySystem & ents, tdt::uint id, ATTACK\_TYPE type )

Changes the attack type of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new attack type.

Definition at line 80 of file CombatHelper.cpp.

### 5.3.2.10 void CombatHelper::set\_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real cd )

Changes the cooldown (minimal time between attacks) of a given entity.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new cooldown value.

Definition at line 64 of file CombatHelper.cpp.

### 5.3.2.11 void CombatHelper::set\_dmg\_range ( EntitySystem & ents, tdt::uint id, tdt::uint min, tdt::uint max )

Changes the damage range (min damage, max damage) that a given entity can deal when attacking.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>Minimal</i>	damage value.
<i>Maximal</i>	damage value.

Definition at line 40 of file CombatHelper.cpp.

### 5.3.2.12 void CombatHelper::set\_projectile\_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )

Sets the projectile table used when a given entity shoots.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>Name</i>	of the projectile table.

Definition at line 108 of file CombatHelper.cpp.

### 5.3.2.13 void CombatHelper::set\_range ( EntitySystem & ents, tdt::uint id, tdt::real range )

Changes the attack range of a given entity.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new attack range.

Definition at line 24 of file CombatHelper.cpp.

5.3.2.14 void `CombatHelper::set_target` ( `EntitySystem & ents`, `tdt::uint id`, `tdt::uint val` )

Changes the target of a given entity's attack.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	containing the entity and it's target.
<i>ID</i>	of the entity.
<i>ID</i>	of the target.

Definition at line 8 of file CombatHelper.cpp.

## 5.4 CommandHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the command component.

### Functions

- void `set_command` (`EntitySystem &`, `tdt::uint`, `COMMAND_TYPE`, `bool=true`)  
*Sets the bit value of a given command for a given entity.*
- bool `test_command` (`EntitySystem &`, `tdt::uint`, `COMMAND_TYPE`)  
*Returns true if a given entity responds to a given type of command, false otherwise.*
- void `command_to_mine` (`EntitySystem &`, `SelectionBox &`)  
*Commands the miner with the smallest task queue (if any) to mine all selected mineable entities.*
- void `command_to_attack` (`EntitySystem &`, `SelectionBox &`)  
*Commands the combat unit with the smallest task queue (if any) to attack a selected enemy.*
- void `command_to_reposition` (`EntitySystem &`, `Ogre::Real`, `Ogre::Real`)  
*Commands the unit with the smallest task queue (if any) to move to a given position.*
- void `command_to_return_gold` (`EntitySystem &`, `CombatSystem &`)  
*Commands all miners that have gold on them to return it to the nearest gold vault.*
- void `command_to_fall_back` (`EntitySystem &`)  
*Commands all free units created at a spawner to return back.*

### 5.4.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the command component.

### 5.4.2 Function Documentation

5.4.2.1 void `CommandHelper::command_to_attack` ( `EntitySystem & ents`, `SelectionBox & selection` )

Commands the combat unit with the smallest task queue (if any) to attack a selected enemy.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>Selection</i>	box used to select the enemy.

Definition at line 70 of file CommandHelper.cpp.

#### 5.4.2.2 void CommandHelper::command\_to\_fall\_back ( EntitySystem & ents )

Commands all free units created at a spawner to return back.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
------------------	---

Definition at line 170 of file CommandHelper.cpp.

#### 5.4.2.3 void CommandHelper::command\_to\_mine ( EntitySystem & ents, SelectionBox & selection )

Commands the miner with the smallest task queue (if any) to mine all selected mineable entities.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>Selection</i>	box used to select the mineable entities.

Definition at line 25 of file CommandHelper.cpp.

#### 5.4.2.4 void CommandHelper::command\_to\_reposition ( EntitySystem & ents, Ogre::Real x, Ogre::Real y )

Commands the unit with the smallest task queue (if any) to move to a given position.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>X</i>	coordinate of the position.
<i>Z</i>	coordinate of the position.

Definition at line 109 of file CommandHelper.cpp.

#### 5.4.2.5 void CommandHelper::command\_to\_return\_gold ( EntitySystem & ents, CombatSystem & combat )

Commands all miners that have gold on them to return it to the nearest gold vault.



## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>Combat</i>	system used to search for deposits.

Definition at line 146 of file CommandHelper.cpp.

**5.4.2.6** `void CommandHelper::set_command ( EntitySystem & ents, tdt::uint id, COMMAND_TYPE command, bool val = true )`

Sets the bit value of a given command for a given entity.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>ID</i>	of the entity.
<i>Type</i>	of the command.
<i>The</i>	new bit value, true = entity responds to the command type, false = entity ignores the command type.

Definition at line 9 of file CommandHelper.cpp.

**5.4.2.7** `bool CommandHelper::test_command ( EntitySystem & ents, tdt::uint id, COMMAND_TYPE command )`

Returns true if a given entity responds to a given type of command, false otherwise.

## Parameters

<i>Reference</i>	to the entity system containing the entity.
<i>ID</i>	of the entity.
<i>Type</i>	of the command.

Definition at line 16 of file CommandHelper.cpp.

## 5.5 ConstructorHelper Namespace Reference

Namespace containing auxiliary functions that help with constructor component management.

### Functions

- void `set_blueprint (EntitySystem &, tdt::uint, const std::string &)`  
Sets the name of the blueprint table that handles the construction of a given entity.
- const std::string & `get_blueprint (EntitySystem &, tdt::uint)`  
Returns the name of the blueprint table that handles the construction of a given entity.
- void `call (EntitySystem &, tdt::uint)`  
Calls the blueprint table that handles the construction of a given entity.

### 5.5.1 Detailed Description

Namespace containing auxiliary functions that help with constructor component management.

### 5.5.2 Function Documentation

#### 5.5.2.1 void ConstructorHelper::call ( EntitySystem & *ents*, tdt::uint *id* )

Calls the blueprint table that handles the construction of a given entity.

Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 22 of file ConstructorHelper.cpp.

#### 5.5.2.2 const std::string & ConstructorHelper::get\_blueprint ( EntitySystem & *ents*, tdt::uint *id* )

Returns the name of the blueprint table that handles the construction of a given entity.

Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 13 of file ConstructorHelper.cpp.

#### 5.5.2.3 void ConstructorHelper::set\_blueprint ( EntitySystem & *ents*, tdt::uint *id*, const std::string & *val* )

Sets the name of the blueprint table that handles the construction of a given entity.

Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new blueprint name.

Definition at line 6 of file ConstructorHelper.cpp.

## 5.6 CounterHelper Namespace Reference

Namespace containing auxiliary functions that help with counter component management.

## Functions

- bool [increment](#) ([EntitySystem](#) &, tdt::uint)  
*Increments the counter of a given entity and returns true if the counter has reached the max value, false otherwise.*
- bool [decrement](#) ([EntitySystem](#) &, tdt::uint)  
*Decrements the counter of a given entity and returns true if the counter has reached the max value, false otherwise.*
- void [set\\_curr\\_value](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the current value of the counter of a given entity.*
- tdt::uint [get\\_curr\\_value](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the current value of the counter of a given entity.*
- void [set\\_max\\_value](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the max value of the counter of a given entity.*
- tdt::uint [get\\_max\\_value](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the max value of the counter of a given entity.*

### 5.6.1 Detailed Description

Namespace containing auxiliary functions that help with counter component management.

### 5.6.2 Function Documentation

#### 5.6.2.1 bool CounterHelper::decrement ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Decrements the counter of a given entity and returns true if the counter has reached the max value, false otherwise.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 17 of file CounterHelper.cpp.

#### 5.6.2.2 tdt::uint CounterHelper::get\_curr\_value ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the current value of the counter of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 36 of file CounterHelper.cpp.

#### 5.6.2.3 tdt::uint CounterHelper::get\_max\_value ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the max value of the counter of a given entity.

## Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 52 of file CounterHelper.cpp.

#### 5.6.2.4 `bool CounterHelper::increment ( EntitySystem & ents, tdt::uint id )`

Increments the counter of a given entity and returns true if the counter has reached the max value, false otherwise.

## Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 5 of file CounterHelper.cpp.

#### 5.6.2.5 `void CounterHelper::set_curr_value ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the current value of the counter of a given entity.

## Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new counter value.

Definition at line 29 of file CounterHelper.cpp.

#### 5.6.2.6 `void CounterHelper::set_max_value ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the max value of the counter of a given entity.

## Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 45 of file CounterHelper.cpp.

## 5.7 DestructorHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the destructor component.

## Functions

- void [set\\_blueprint](#) ([EntitySystem](#) &, tdt::uint, const std::string &)  
*Sets the name of the table that contains the "dtor" function which get's called when a given entity is destroyed.*
- const std::string & [get\\_blueprint](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the name of the table that contains the "dtor" function which get's called when a given entity is destroyed.*
- void [destroy](#) ([EntitySystem](#) &, tdt::uint, bool=false, tdt::uint=Component::NO\_ENTITY)  
*Destroys a given entity and if possible calls it's destructor.*

### 5.7.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the destructor component.

### 5.7.2 Function Documentation

**5.7.2.1** void DestructorHelper::destroy ( [EntitySystem](#) & *ents*, tdt::uint *id*, bool *supress\_dtor* = false, tdt::uint *killer* = Component::NO\_ENTITY )

Destroys a given entity and if possible calls it's destructor.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.
<i>If</i>	true, the destructor won't be called.
<i>ID</i>	of the killer (if any).

Definition at line 25 of file DestructorHelper.cpp.

**5.7.2.2** const std::string & DestructorHelper::get\_blueprint ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the name of the table that contains the "dtor" function which get's called when a given entity is destroyed.

#### Parameters

<a href="#">EntitySystem</a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 14 of file DestructorHelper.cpp.

**5.7.2.3** void DestructorHelper::set\_blueprint ( [EntitySystem](#) & *ents*, tdt::uint *id*, const std::string & *val* )

Sets the name of the table that contains the "dtor" function which get's called when a given entity is destroyed.

## Parameters

<i>EntitySystem</i>	containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new blueprint table's name.

Definition at line 7 of file DestructorHelper.cpp.

## 5.8 EventHandlerHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the event handler component.

## Functions

- void `set_handler` (`EntitySystem` &, `tdt::uint`, `const std::string` &)  
*Sets the name of the table that contains event the handler function of a given entity.*
- `const std::string` & `get_handler` (`EntitySystem` &, `tdt::uint`)  
*Returns the name of the table that contains event the handler function of a given entity.*
- bool `can_handle` (`EntitySystem` &, `tdt::uint`, `EVENT_TYPE`)  
*Returns true if a given entity can handle a given event.*
- void `add_possible_event` (`EntitySystem` &, `tdt::uint`, `EVENT_TYPE`)  
*Adds a given event into the list of possible events of a given entity.*
- void `delete_possible_event` (`EntitySystem` &, `tdt::uint`, `EVENT_TYPE`)  
*Removes a given event from the list of possible events of a given entity.*

### 5.8.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the event handler component.

### 5.8.2 Function Documentation

#### 5.8.2.1 void EventHandlerHelper::add\_possible\_event ( EntitySystem & ents, tdt::uint id, EVENT\_TYPE val )

Adds a given event into the list of possible events of a given entity.

## Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	type of the event.

Definition at line 32 of file EventHandlerHelper.cpp.

#### 5.8.2.2 bool EventHandlerHelper::can\_handle ( EntitySystem & ents, tdt::uint id, EVENT\_TYPE val )

Returns true if a given entity can handle a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	type of the event.

Definition at line 23 of file EventHandlerHelper.cpp.

#### 5.8.2.3 void EventHandlerHelper::delete\_possible\_event ( EntitySystem & ents, tdt::uint id, EVENT\_TYPE val )

Removes a given event from the list of possible events of a given entity.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	type of the event.

Definition at line 39 of file EventHandlerHelper.cpp.

#### 5.8.2.4 const std::string & EventHandlerHelper::get\_handler ( EntitySystem & ents, tdt::uint id )

Returns the name of the table that contains event the handler function of a given entity.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file EventHandlerHelper.cpp.

#### 5.8.2.5 void EventHandlerHelper::set\_handler ( EntitySystem & ents, tdt::uint id, const std::string & val )

Sets the name of the table that contains event the handler function of a given entity.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new handler table name.

Definition at line 5 of file EventHandlerHelper.cpp.

## 5.9 EventHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the event component.

### Functions

- void [set\\_event\\_type](#) (EntitySystem &, tdt::uint, EVENT\_TYPE)  
*Sets the type of a given event.*
- EVENT\_TYPE [get\\_event\\_type](#) (EntitySystem &, tdt::uint)  
*Returns the type of a given event.*
- void [set\\_target](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the target entity (the subject of the event) of a given event.*
- tdt::uint [get\\_target](#) (EntitySystem &, tdt::uint)  
*Returns the target entity (the subject of the event) of a given event.*
- void [set\\_radius](#) (EntitySystem &, tdt::uint, tdt::real)  
*Sets the radius handlers have to be in in order to be able to handle a given event.*
- tdt::real [get\\_radius](#) (EntitySystem &, tdt::uint)  
*Returns the radius handlers have to be in in order to be able to handle a given event.*
- void [set\\_active](#) (EntitySystem &, tdt::uint, bool=true)  
*Sets the activity state of a given event.*
- bool [is\\_active](#) (EntitySystem &, tdt::uint)  
*Returns true if a given event is active, false otherwise.*
- void [set\\_event\\_handler](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the entity that handles a given event.*
- tdt::uint [get\\_event\\_handler](#) (EntitySystem &, tdt::uint)  
*Returns the entity that handles a given event.*

### 5.9.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the event component.

### 5.9.2 Function Documentation

#### 5.9.2.1 std::size\_t EventHelper::get\_event\_handler ( EntitySystem & ents, tdt::uint id )

Returns the entity that handles a given event.

#### Parameters

<i>Entity</i>	system containing both the entity and the event.
<i>ID</i>	of the event.

Definition at line 76 of file EventHelper.cpp.



#### 5.9.2.2 EVENT\_TYPE EventHelper::get\_event\_type ( EntitySystem & ents, tdt::uint id )

Returns the type of a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.

Definition at line 12 of file EventHelper.cpp.

#### 5.9.2.3 Ogre::Real EventHelper::get\_radius ( EntitySystem & ents, tdt::uint id )

Returns the radius handlers have to be in in order to be able to handle a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.

Definition at line 44 of file EventHelper.cpp.

#### 5.9.2.4 std::size\_t EventHelper::get\_target ( EntitySystem & ents, tdt::uint id )

Returns the target entity (the subject of the event) of a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.

Definition at line 28 of file EventHelper.cpp.

#### 5.9.2.5 bool EventHelper::is\_active ( EntitySystem & ents, tdt::uint id )

Returns true if a given event is active, false otherwise.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.

Definition at line 60 of file EventHelper.cpp.

#### 5.9.2.6 void EventHelper::set\_active ( EntitySystem & ents, tdt::uint id, bool val = true )

Sets the activity state of a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.
<i>ID</i>	for active, false for inactive.

Definition at line 53 of file EventHelper.cpp.

#### 5.9.2.7 void EventHelper::set\_event\_handler ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the entity that handles a given event.

##### Parameters

<i>Entity</i>	system containing both the entity and the event.
<i>ID</i>	of the event.
<i>ID</i>	of the handling entity.

Definition at line 69 of file EventHelper.cpp.

#### 5.9.2.8 void EventHelper::set\_event\_type ( EntitySystem & ents, tdt::uint id, EVENT\_TYPE val )

Sets the type of a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.
<i>The</i>	new type.

Definition at line 5 of file EventHelper.cpp.

#### 5.9.2.9 void EventHelper::set\_radius ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the radius handlers have to be in in order to be able to handle a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.
<i>The</i>	new radius.

Definition at line 37 of file EventHelper.cpp.

#### 5.9.2.10 void EventHelper::set\_target ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the target entity (the subject of the event) of a given event.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the event.
<i>ID</i>	of the target.

Definition at line 21 of file EventHelper.cpp.

## 5.10 ExperienceValueHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the experience value component.

### Functions

- void [set](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the experience value a given entity is worth.*
- tdt::uint [get](#) (EntitySystem &, tdt::uint)  
*Returns the experience value a given entity is worth.*
- void [increase](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Increases the experience value a given entity is worth by a given value.*
- void [decrease](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Decreases the experience value a given entity is worth by a given value.*

### 5.10.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the experience value component.

### 5.10.2 Function Documentation

#### 5.10.2.1 void ExperienceValueHelper::decrease ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Decreases the experience value a given entity is worth by a given value.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	value to decrease by.

Definition at line 28 of file ExperienceValueHelper.cpp.

#### 5.10.2.2 `tdt::uint ExperienceValueHelper::get ( EntitySystem & ents, tdt::uint id )`

Returns the experience value a given entity is worth.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file ExperienceValueHelper.cpp.

#### 5.10.2.3 `void ExperienceValueHelper::increase ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Increases the experience value a given entity is worth by a given value.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	value to increase by.

Definition at line 21 of file ExperienceValueHelper.cpp.

#### 5.10.2.4 `void ExperienceValueHelper::set ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the experience value a given entity is worth.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new experience value.

Definition at line 5 of file ExperienceValueHelper.cpp.

## 5.11 ExplosionHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the explosion component.

### Functions

- void [set\\_delta](#) ([EntitySystem](#) &, tdt::uint, tdt::real)

- Sets the value by which a given entity's explosion radius is updated on each of the graphics system updates.*
- `tdt::real get_delta (EntitySystem &, tdt::uint)`  
*Returns the value by which a given entity's explosion radius is updated on each of the graphics system updates.*
- `void set_max_radius (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the radius after which a given entity with an explosion component gets deleted.*
- `tdt::real get_max_radius (EntitySystem &, tdt::uint)`  
*Returns the radius after which a given entity with an explosion component gets deleted.*
- `tdt::real get_curr_radius (EntitySystem &, tdt::uint)`  
*Returns the current radius of a given entity's explosion component.*
- `void increase_curr_radius (EntitySystem &, tdt::uint, tdt::real)`  
*Increases the radius of a given entity's explosion component by a given value.*

### 5.11.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the explosion component.

### 5.11.2 Function Documentation

#### 5.11.2.1 `tdt::real ExplosionHelper::get_curr_radius ( EntitySystem & ents, tdt::uint id )`

Returns the current radius of a given entity's explosion component.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 37 of file ExplosionHelper.cpp.

#### 5.11.2.2 `tdt::real ExplosionHelper::get_delta ( EntitySystem & ents, tdt::uint id )`

Returns the value by which a given entity's explosion radius is updated on each of the graphics system updates.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file ExplosionHelper.cpp.

#### 5.11.2.3 `tdt::real ExplosionHelper::get_max_radius ( EntitySystem & ents, tdt::uint id )`

Returns the radius after which a given entity with an explosion component gets deleted.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 28 of file ExplosionHelper.cpp.

#### 5.11.2.4 void ExplosionHelper::increase\_curr\_radius ( EntitySystem & ents, tdt::uint id, tdt::real val )

Increases the radius of a given entity's explosion component by a given value.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	value to increase by.

Definition at line 46 of file ExplosionHelper.cpp.

#### 5.11.2.5 void ExplosionHelper::set\_delta ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the value by which a given entity's explosion radius is updated on each of the graphics system updates.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new delta value.

Definition at line 5 of file ExplosionHelper.cpp.

#### 5.11.2.6 void ExplosionHelper::set\_max\_radius ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the radius after which a given entity with an explosion component gets deleted.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new max radius value.

Definition at line 21 of file ExplosionHelper.cpp.

## 5.12 FactionHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the faction component.

### Functions

- void `set_faction` (`EntitySystem` &, `tdt::uint`, `FACTION`)  
*Changes the FACTION of a given entity.*
- `FACTION` `get_faction` (`EntitySystem` &, `tdt::uint`)  
*Returns the FACTION of a given entity.*
- const `std::string` & `get_faction_name` (`EntitySystem` &, `tdt::uint`)  
*Returns the FACTION name (a string) of a given entity.*

### 5.12.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the faction component.

### 5.12.2 Function Documentation

#### 5.12.2.1 `FACTION` `FactionHelper::get_faction` ( `EntitySystem` & *ents*, `tdt::uint` *id* )

Returns the FACTION of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 12 of file `FactionHelper.cpp`.

#### 5.12.2.2 `const std::string` & `FactionHelper::get_faction_name` ( `EntitySystem` & *ents*, `tdt::uint` *id* )

Returns the FACTION name (a string) of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 21 of file `FactionHelper.cpp`.

#### 5.12.2.3 `void` `FactionHelper::set_faction` ( `EntitySystem` & *ents*, `tdt::uint` *id*, `FACTION` *val* )

Changes the FACTION of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new faction.

Definition at line 5 of file FactionHelper.cpp.

## 5.13 GoldHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the gold component.

### Functions

- void [set\\_curr\\_gold](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the gold value a given entity has, does not check if the new value is smaller than the limit, use add\_gold if that's needed.*
- tdt::uint [get\\_curr\\_gold](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the gold value a given entity has.*
- void [set\\_max\\_gold](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the limit of gold that a given entity can have.*
- tdt::uint [get\\_max\\_gold](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the limit of gold that a given entity can have.*
- tdt::uint [add\\_gold](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Adds a given gold value to a given entity up to it's gold limit, returns the amount of gold that superceeded the gold limit and thus wasn't added.*
- tdt::uint [sub\\_gold](#) ([EntitySystem](#) &, tdt::uint, tdt::uint, bool=true)  
*Removes a given amount of gold from a given entity, but does not subtract past zero.*
- tdt::uint [transfer\\_all\\_gold](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Transfers all gold of an entity to another entity while keeping the (0, max) range in mind.*
- bool [gold\\_full](#) ([EntitySystem](#) &, tdt::uint)  
*Returns true if a given entity's gold storage is full, false otherwise.*
- void [register\\_transaction\\_](#) ([EntitySystem](#) &, [GoldComponent](#) &, tdt::uint, tdt::uint, bool=true)

### 5.13.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the gold component.

### 5.13.2 Function Documentation

#### 5.13.2.1 tdt::uint GoldHelper::add\_gold ( [EntitySystem](#) & *ents*, tdt::uint *id*, tdt::uint *val* )

Adds a given gold value to a given entity up to it's gold limit, returns the amount of gold that superceeded the gold limit and thus wasn't added.



## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>Amount</i>	of gold to add.

Definition at line 50 of file GoldHelper.cpp.

#### 5.13.2.2 tdt::uint GoldHelper::get\_curr\_gold ( EntitySystem & ents, tdt::uint id )

Returns the gold value a given entity has.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 22 of file GoldHelper.cpp.

#### 5.13.2.3 tdt::uint GoldHelper::get\_max\_gold ( EntitySystem & ents, tdt::uint id )

Returns the limit of gold that a given entity can have.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 41 of file GoldHelper.cpp.

#### 5.13.2.4 bool GoldHelper::gold\_full ( EntitySystem & ents, tdt::uint id )

Returns true if a given entity's gold storage is full, false otherwise.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 116 of file GoldHelper.cpp.

#### 5.13.2.5 void GoldHelper::set\_curr\_gold ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the gold value a given entity has, does not check if the new value is smaller than the limit, use add\_gold if that's needed.

## Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new gold value.

Definition at line 8 of file GoldHelper.cpp.

5.13.2.6 `void GoldHelper::set_max_gold ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the limit of gold that a given entity can have.

## Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new limit.

Definition at line 31 of file GoldHelper.cpp.

5.13.2.7 `tdt::uint GoldHelper::sub_gold ( EntitySystem & ents, tdt::uint id, tdt::uint val, bool reg = true )`

Removes a given amount of gold from a given entity, but does not subtract past zero.

Returns amount of gold that could not be removed (remainder from the given amount after subtracting).

## Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>Amount</i>	of gold to subtract.
<i>If</i>	false, the transaction won't be registered (used when called from <a href="#">Player</a> );

Definition at line 70 of file GoldHelper.cpp.

5.13.2.8 `tdt::uint GoldHelper::transfer_all_gold ( EntitySystem & ents, tdt::uint id_from, tdt::uint id_to )`

Transfers all gold of an entity to another entity while keeping the (0, max) range in mind.

Returns amount of gold actually transfered.

## Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the sender entity.
<i>ID</i>	of the receiver entity.

Definition at line 88 of file GoldHelper.cpp.

## 5.14 GraphicsHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the graphics component.

### Enumerations

- enum [PLANE](#) { **X** = 0, **Y**, **Z** }  
*Enum representing the planes in a 3D space.*

### Functions

- void [set\\_mesh](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the model of a given entity.*
- const std::string & [get\\_mesh](#) (EntitySystem &, tdt::uint)  
*Returns the name of the mesh of a given entity.*
- void [set\\_material](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the name of the material a given entity is using.*
- const std::string & [get\\_material](#) (EntitySystem &, tdt::uint)  
*Returns the name of the material a given entity is using.*
- void [set\\_visible](#) (EntitySystem &, tdt::uint, bool)  
*Sets the visibility status of a given entity.*
- bool [is\\_visible](#) (EntitySystem &, tdt::uint)  
*Returns true if a given entity is visible, false otherwise.*
- void [set\\_manual\\_scaling](#) (EntitySystem &, tdt::uint, bool)  
*Sets the manual scaling status of a given entity.*
- bool [get\\_manual\\_scaling](#) (EntitySystem &, tdt::uint)  
*Returns true if a given entity's model has explicit dimensions or false if the dimensions of it's mesh are used.*
- void [set\\_scale](#) (EntitySystem &, tdt::uint, const Ogre::Vector3 &)  
*Changes the dimensions of a given entity (requires manual\_scaling to be true).*
- const Ogre::Vector3 & [get\\_scale](#) (EntitySystem &, tdt::uint)  
*Returns the scale (the dimensions) of a given entity (requires manual\_scaling to be true).*
- void [look\\_at](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Rotates a given entity so that it faces another one.*
- void [rotate](#) (EntitySystem &, tdt::uint, tdt::real, [PLANE](#)=[PLANE::Y](#))  
*Rotates a given entity by a given amount of radians.*
- const Ogre::AxisAlignedBox & [get\\_bounds](#) (EntitySystem &, tdt::uint)  
*Returns a given entity's bounding box.*
- bool [collide](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Returns true if two given entities collide, false otherwise.*
- void [init\\_graphics\\_component](#) (EntitySystem &, Ogre::SceneManager &, tdt::uint)  
*Initializes the graphics component of a manually created entity by loading it's model into an Ogre::Entity and bounding it to a scene node.*
- void [set\\_query\\_flags](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the flags used for queries (like in [CombatSystem::in\\_sight](#)) of a given entity.*
- tdt::uint [get\\_query\\_flags](#) (EntitySystem &, tdt::uint)  
*Return the flags used for queries (like in [CombatSystem::in\\_sight](#)) of a given entity.*
- void [apply\\_scale](#) (EntitySystem &, tdt::uint)  
*Applies scale to a given entity's Ogre::Node if the entity has manual scaling enabled (the scale is a part of the component).*

### 5.14.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the graphics component.

### 5.14.2 Enumeration Type Documentation

#### 5.14.2.1 enum GraphicsHelper::PLANE [strong]

Enum representing the planes in a 3D space.

Definition at line 16 of file GraphicsHelper.hpp.

### 5.14.3 Function Documentation

#### 5.14.3.1 void GraphicsHelper::apply\_scale ( EntitySystem & ents, tdt::uint id )

Applies scale to a given entity's Ogre::Node if the entity has manual scaling enabled (the scale is a part of the component).

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 211 of file GraphicsHelper.cpp.

#### 5.14.3.2 bool GraphicsHelper::collide ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )

Returns true if two given entities collide, false otherwise.

Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity.

Definition at line 145 of file GraphicsHelper.cpp.

#### 5.14.3.3 const Ogre::AxisAlignedBox & GraphicsHelper::get\_bounds ( EntitySystem & ents, tdt::uint id )

Returns a given entity's bounding box.

Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of th entity.

**Note**

The entity has to have a [GraphicsComponent](#), because collision detection is done using Ogre's bounding boxes.

Definition at line 136 of file GraphicsHelper.cpp.

**5.14.3.4 bool GraphicsHelper::get\_manual\_scaling ( EntitySystem & ents, tdt::uint id )**

Returns true if a given entity's model has explicit dimensions or false if the dimensions of it's mesh are used.

**Parameters**

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.

Definition at line 72 of file GraphicsHelper.cpp.

**5.14.3.5 const std::string & GraphicsHelper::get\_material ( EntitySystem & ents, tdt::uint id )**

Returns the name of the material a given entity is using.

**Parameters**

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.

Definition at line 34 of file GraphicsHelper.cpp.

**5.14.3.6 const std::string & GraphicsHelper::get\_mesh ( EntitySystem & ents, tdt::uint id )**

Returns the name of the mesh of a given entity.

**Parameters**

<i>Reference</i>	to the entity system that contains components.
------------------	--

Definition at line 12 of file GraphicsHelper.cpp.

**5.14.3.7 tdt::uint GraphicsHelper::get\_query\_flags ( EntitySystem & ents, tdt::uint id )**

Return the flags used for queries (like in [CombatSystem::in\\_sight](#)) of a given entity.

**Parameters**

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 202 of file GraphicsHelper.cpp.

#### 5.14.3.8 `const Ogre::Vector3 & GraphicsHelper::get_scale ( EntitySystem & ents, tdt::uint id )`

Returns the scale (the dimensions) of a given entity (requires manual\_scaling to be true).

##### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.

Definition at line 88 of file GraphicsHelper.cpp.

#### 5.14.3.9 `void GraphicsHelper::init_graphics_component ( EntitySystem & ents, Ogre::SceneManager & scene, tdt::uint id )`

Initializes the graphics component of a manually created entity by loading it's model into an Ogre::Entity and bounding it to a scene node.

##### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.

Definition at line 150 of file GraphicsHelper.cpp.

#### 5.14.3.10 `bool GraphicsHelper::is_visible ( EntitySystem & ents, tdt::uint id )`

Returns true if a given entity is visible, false otherwise.

##### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.

Definition at line 56 of file GraphicsHelper.cpp.

#### 5.14.3.11 `void GraphicsHelper::look_at ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )`

Rotates a given entity so that it faces another one.

##### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity.

Definition at line 99 of file GraphicsHelper.cpp.

5.14.3.12 `void GraphicsHelper::rotate ( EntitySystem & ents, tdt::uint id, tdt::real delta, PLANE plane = PLANE : : Y )`

Rotates a given entity by a given amount of radians.

#### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>Rotation</i>	angle in radians.

#### Note

Ogre3D has conversion functions.

Definition at line 114 of file GraphicsHelper.cpp.

5.14.3.13 `void GraphicsHelper::set_manual_scaling ( EntitySystem & ents, tdt::uint id, bool val )`

Sets the manual scaling status of a given entity.

(i.e. if the model should use dimensions stored in the mesh file or has them explicitly set.)

#### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>True</i>	for manual scaling, false for using the scale of the mesh.

#### Note

Requires a call to `init_graphics_component` to take effect.

Definition at line 65 of file GraphicsHelper.cpp.

5.14.3.14 `void GraphicsHelper::set_material ( EntitySystem & ents, tdt::uint id, const std::string & material )`

Sets the name of the material a given entity is using.

#### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>Name</i>	of the entity.

**Note**

Requires a call to `init_graphcis_component` to take effect.

Definition at line 23 of file `GraphicsHelper.cpp`.

**5.14.3.15 void GraphicsHelper::set\_mesh ( EntitySystem & ents, tdt::uint id, const std::string & mesh )**

Sets the model of a given entity.

**Parameters**

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>Name</i>	of the new mesh.

**Note**

Requires a call to `init_graphics_component` to take effect.

Definition at line 5 of file `GraphicsHelper.cpp`.

**5.14.3.16 void GraphicsHelper::set\_query\_flags ( EntitySystem & ents, tdt::uint id, tdt::uint val )**

Sets the flags used for queries (like in [CombatSystem::in\\_sight](#)) of a given entity.

**Parameters**

<i>ID</i>	of the entity.
<i>The</i>	new query flags.

Definition at line 195 of file `GraphicsHelper.cpp`.

**5.14.3.17 void GraphicsHelper::set\_scale ( EntitySystem & ents, tdt::uint id, const Ogre::Vector3 & val )**

Changes the dimensions of a given entity (requires `manual_scaling` to be true).

**Parameters**

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>The</i>	new scale value.

**Note**

Requires a call to `init_graphics_component` to take effect.

Definition at line 81 of file `GraphicsHelper.cpp`.



#### 5.14.3.18 void GraphicsHelper::set\_visible ( EntitySystem & ents, tdt::uint id, bool val )

Sets the visibility status of a given entity.

##### Parameters

<i>Reference</i>	to the entity system that contains components.
<i>ID</i>	of the entity.
<i>True</i>	for visible, false for invisible.

Definition at line 45 of file GraphicsHelper.cpp.

## 5.15 GridNodeHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the grid node component.

### Functions

- const std::array< tdt::uint, GridNodeComponent::neighbour\_count > & [get\\_neighbours](#) (EntitySystem &, tdt::uint)  
*Returns an array containing IDs of all neighbours of a given node.*
- bool [is\\_free](#) (EntitySystem &, tdt::uint)  
*Returns true if the given node is free (duh...), false otherwise.*
- bool [area\\_free](#) (EntitySystem &, tdt::uint, tdt::uint=1)  
*Returns true if a given area (specified by a center node and a radius) is free.*
- void [set\\_free](#) (EntitySystem &, tdt::uint, bool)  
*Sets the free status of a given node.*
- void [set\\_free\\_selected](#) (EntitySystem &, SelectionBox &, bool)  
*Applies the GridSystem::set\_free method to all currently selected nodes.*
- std::tuple< tdt::uint, tdt::uint > [get\\_board\\_coords](#) (EntitySystem &, tdt::uint)  
*Returns the board relative coordinates (row & column) of a given node.*
- void [set\\_resident](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the resident of a given node.*
- tdt::uint [get\\_resident](#) (EntitySystem &, tdt::uint)  
*Returns the resident of a given node.*
- tdt::uint [get\\_manhattan\\_distance](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Returns the manhattan (definition available on Wikipedia...) distance between two nodes.*
- tdt::uint [get\\_node\\_in\\_dir](#) (EntitySystem &, tdt::uint, int)  
*Returns the closest node in a given direction.*
- void [set\\_portal\\_neighbour](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the portal node linked to this node.*

### 5.15.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the grid node component.

### 5.15.2 Function Documentation

#### 5.15.2.1 bool GridNodeHelper::area\_free ( EntitySystem & ents, tdt::uint center, tdt::uint radius = 1 )

Returns true if a given area (specified by a center node and a radius) is free.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the nodes in the area.
<i>ID</i>	of the center node.
<i>Radius</i>	of the area.

## Note

This counts also walkthrough buildings, as it's only used for building placing and not pathfinding.

Definition at line 27 of file GridNodeHelper.cpp.

5.15.2.2 `std::tuple< tdt::uint, tdt::uint > GridNodeHelper::get_board_coords ( EntitySystem & ents, tdt::uint id )`

Returns the board relative coordinates (row & column) of a given node.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the node.
<i>ID</i>	of the node.

Definition at line 68 of file GridNodeHelper.cpp.

5.15.2.3 `tdt::uint GridNodeHelper::get_manhattan_distance ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )`

Returns the manhattan (definition available on Wikipedia...) distance between two nodes.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the nodes.
<i>ID</i>	of the source node.
<i>ID</i>	of the target node.

Definition at line 99 of file GridNodeHelper.cpp.

5.15.2.4 `const std::array< tdt::uint, GridNodeComponent::neighbour_count > & GridNodeHelper::get_neighbours ( EntitySystem & ents, tdt::uint id )`

Returns an array containing IDs of all neighbours of a given node.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the node.
<i>ID</i>	of the node.

Definition at line 8 of file GridNodeHelper.cpp.

#### 5.15.2.5 tdt::uint GridNodeHelper::get\_node\_in\_dir ( EntitySystem & ents, tdt::uint id, int dir )

Returns the closest node in a given direction.

##### Parameters

<i>EntitySystem</i>	containing the node.
<i>ID</i>	of the entity that is looking for the node.
<i>The</i>	direction represented by the DIRECTION::VAL enum.

Definition at line 110 of file GridNodeHelper.cpp.

#### 5.15.2.6 tdt::uint GridNodeHelper::get\_resident ( EntitySystem & ents, tdt::uint id )

Returns the resident of a given node.

##### Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 90 of file GridNodeHelper.cpp.

#### 5.15.2.7 bool GridNodeHelper::is\_free ( EntitySystem & ents, tdt::uint id )

Returns true if the given node is free (duh...), false otherwise.

##### Parameters

<i>EntitySystem</i>	containing the node.
<i>ID</i>	of the node.

Definition at line 18 of file GridNodeHelper.cpp.

#### 5.15.2.8 void GridNodeHelper::set\_free ( EntitySystem & ents, tdt::uint id, bool val )

Sets the free status of a given node.

##### Parameters

<i>EntitySystem</i>	containing the node.
<i>ID</i>	of the node.
<i>True</i>	for free, false for not-so-free.

Definition at line 46 of file GridNodeHelper.cpp.

#### 5.15.2.9 void GridNodeHelper::set\_free\_selected ( EntitySystem & ents, SelectionBox & box, bool val )

Applies the GridSystem::set\_free method to all currently selected nodes.

##### Parameters

<a href="#">EntitySystem</a>	containing the nodes.
<i>Reference</i>	to the selection box that selected the nodes.
<i>True</i>	for free, false for not-so-free.

Definition at line 62 of file GridNodeHelper.cpp.

#### 5.15.2.10 void GridNodeHelper::set\_portal\_neighbour ( EntitySystem & ents, tdt::uint id, tdt::uint portal )

Sets the portal node linked to this node.

##### Parameters

<a href="#">EntitySystem</a>	containing the node.
<i>ID</i>	of this node.
<i>ID</i>	of the portal node.

Definition at line 125 of file GridNodeHelper.cpp.

#### 5.15.2.11 void GridNodeHelper::set\_resident ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the resident of a given node.

(Resident is an entity that is causing the node to be not free - like a wall, building etc.)

##### Parameters

<a href="#">EntitySystem</a>	containing the node and the resident.
<i>ID</i>	of the node.
<i>ID</i>	of the resident.

Definition at line 77 of file GridNodeHelper.cpp.

## 5.16 HealthHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the health component.

### Functions

- void [set\\_health](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)

- Sets the health of a given entity without any regard to it's maximal health.*
- `tdt::uint get_health (EntitySystem &, tdt::uint)`  
*Returns the current health amount of a given entity.*
- `void add_health (EntitySystem &, tdt::uint, tdt::uint)`  
*Increases the current health amount of an entity by a given amount up to the maximum value stored in it's [HealthComponent](#).*
- `void sub_health (EntitySystem &, tdt::uint, tdt::uint, bool=false)`  
*Subtracts a given amount from the current health of an entity, taking it's defense into account by default.*
- `void heal (EntitySystem &, tdt::uint)`  
*Sets the current health amount of an entity to it's maximum value.*
- `void buff (EntitySystem &, tdt::uint, tdt::uint)`  
*Increases the current and maximum health amount of an entity by a given value.*
- `void debuff (EntitySystem &, tdt::uint, tdt::uint)`  
*Reduces the current and maximum health amount of an entity by a given value.*
- `void set_regen (EntitySystem &, tdt::uint, tdt::uint)`  
*Sets the regeneration value of a given entity.*
- `tdt::uint get_regen (EntitySystem &, tdt::uint)`  
*Returns the regeneration value of a given entity.*
- `void set_alive (EntitySystem &, tdt::uint, bool)`  
*Allows to set the health status of an entity without adding/subing health.*
- `bool is_alive (EntitySystem &, tdt::uint)`  
*Returns true if a given entity is alive, false otherwise.*
- `void set_defense (EntitySystem &, tdt::uint, tdt::uint)`  
*Sets the amount of defense a given entity has to a given (absolute) amount.*
- `tdt::uint get_defense (EntitySystem &, tdt::uint)`  
*Returns the defense of a given entity.*
- `void add_defense (EntitySystem &, tdt::uint, tdt::uint)`  
*Increases the defense of an entity by a given amount.*
- `void sub_defense (EntitySystem &, tdt::uint, tdt::uint)`  
*Reduces the defense of an entity by a given amount.*
- `void ubercharge (EntitySystem &, tdt::uint)`  
*A cheat that sets the health, maximum health and defense to their highest possible values.*

### 5.16.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the health component.

### 5.16.2 Function Documentation

#### 5.16.2.1 `void HealthHelper::add_defense ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Increases the defense of an entity by a given amount.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of defense to be added.

Definition at line 163 of file HealthHelper.cpp.

#### 5.16.2.2 void HealthHelper::add\_health ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Increases the current health amount of an entity by a given amount up to the maximum value stored in it's [HealthComponent](#).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of health to be added. Note: To increase health along with the maximum value, see HealthSystem::buff.

Definition at line 31 of file HealthHelper.cpp.

#### 5.16.2.3 void HealthHelper::buff ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Increases the current and maximum health amount of an entity by a given value.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of health to be added.

Definition at line 79 of file HealthHelper.cpp.

#### 5.16.2.4 void HealthHelper::debuff ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Reduces the current and maximum health amount of an entity by a given value.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of health to be subtracted.

Definition at line 93 of file HealthHelper.cpp.

#### 5.16.2.5 tdt::uint HealthHelper::get\_defense ( EntitySystem & ents, tdt::uint id )

Returns the defense of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 154 of file HealthHelper.cpp.

**5.16.2.6 tdt::uint HealthHelper::get\_health ( EntitySystem & ents, tdt::uint id )**

Returns the current health amount of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 22 of file HealthHelper.cpp.

**5.16.2.7 tdt::uint HealthHelper::get\_regen ( EntitySystem & ents, tdt::uint id )**

Returns the regeneration value of a given entity.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 122 of file HealthHelper.cpp.

**5.16.2.8 void HealthHelper::heal ( EntitySystem & ents, tdt::uint id )**

Sets the current health amount of an entity to it's maximum value.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 66 of file HealthHelper.cpp.

**5.16.2.9 bool HealthHelper::is\_alive ( EntitySystem & ents, tdt::uint id )**

Returns true if a given entity is alive, false otherwise.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 138 of file HealthHelper.cpp.

**5.16.2.10 void HealthHelper::set\_alive ( EntitySystem & ents, tdt::uint id, bool val )**

Allows to set the health status of an entity without adding/subing health.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>True</i>	for "alive" and false for "dead".

Definition at line 131 of file HealthHelper.cpp.

**5.16.2.11 void HealthHelper::set\_defense ( EntitySystem & ents, tdt::uint id, tdt::uint val )**

Sets the amount of defense a given entity has to a given (absolute) amount.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new defense value.

Definition at line 147 of file HealthHelper.cpp.

**5.16.2.12 void HealthHelper::set\_health ( EntitySystem & ents, tdt::uint id, tdt::uint val )**

Sets the health of a given entity without any regard to it's maximal health.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new health value.

Definition at line 7 of file HealthHelper.cpp.

**5.16.2.13 void HealthHelper::set\_regen ( EntitySystem & ents, tdt::uint id, tdt::uint val )**

Sets the regeneration value of a given entity.



## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new regen value.

Definition at line 115 of file HealthHelper.cpp.

#### 5.16.2.14 void HealthHelper::sub\_defense ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Reduces the defense of an entity by a given amount.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of defense to be removed.

Definition at line 170 of file HealthHelper.cpp.

#### 5.16.2.15 void HealthHelper::sub\_health ( EntitySystem & ents, tdt::uint id, tdt::uint val, bool ignore\_armor = false )

Subtracts a given amount from the current health of an entity, taking it's defense into account by default.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Amount</i>	of health to be subtracted.
<i>Optional</i>	boolean indicator, if true, the entity's defense will be ignored, otherwise it will be subtracted from the given amount.

Definition at line 44 of file HealthHelper.cpp.

#### 5.16.2.16 void HealthHelper::ubercharge ( EntitySystem & ents, tdt::uint id )

A cheat that sets the health, maximum health and defense to their highest possible values.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 182 of file HealthHelper.cpp.

## 5.17 HomingHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the homing component.

### Functions

- void [set\\_source](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the ID of the source (entity that shot it) of a given projectile.*
- tdt::uint [get\\_source](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the ID of the entity that shot a given projectile.*
- void [set\\_target](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Changes the target of a given homing projectile.*
- tdt::uint [get\\_target](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the ID of the target of a given homing projectile.*
- void [set\\_dmg](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Changes the damage of a given homing projectile.*
- tdt::uint [get\\_dmg](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the damage value of a given projectile.*

### 5.17.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the homing component.

### 5.17.2 Function Documentation

#### 5.17.2.1 tdt::uint HomingHelper::get\_dmg ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the damage value of a given projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.

Definition at line 44 of file HomingHelper.cpp.

#### 5.17.2.2 tdt::uint HomingHelper::get\_source ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the ID of the entity that shot a given projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.

Definition at line 12 of file HomingHelper.cpp.

#### 5.17.2.3 tdt::uint HomingHelper::get\_target ( EntitySystem & ents, tdt::uint id )

Returns the ID of the target of a given homing projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.

Definition at line 28 of file HomingHelper.cpp.

#### 5.17.2.4 void HomingHelper::set\_dmg ( EntitySystem & ents, tdt::uint id, tdt::uint dmg )

Changes the damage of a given homing projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.
<i>The</i>	new damage value.

Definition at line 37 of file HomingHelper.cpp.

#### 5.17.2.5 void HomingHelper::set\_source ( EntitySystem & ents, tdt::uint id, tdt::uint source )

Sets the ID of the source (entity that shot it) of a given projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.
<i>ID</i>	of the source.

Definition at line 5 of file HomingHelper.cpp.

#### 5.17.2.6 void HomingHelper::set\_target ( EntitySystem & ents, tdt::uint id, tdt::uint target )

Changes the target of a given homing projectile.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the projectile.
<i>ID</i>	of the target.

Definition at line 21 of file HomingHelper.cpp.

## 5.18 InputHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the input component.

### Functions

- void [set\\_input\\_handler](#) ([EntitySystem](#) &, tdt::uint, const std::string &)  
*Changes the name of the table that contains a given entity's input handler.*
- const std::string & [get\\_input\\_handler](#) ([EntitySystem](#) &, tdt::uint)  
*Returns name of the table that contains the input handling function of a given entity.*

### 5.18.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the input component.

### 5.18.2 Function Documentation

#### 5.18.2.1 const std::string & InputHelper::get\_input\_handler ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns name of the table that contains the input handling function of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 12 of file InputHelper.cpp.

#### 5.18.2.2 void InputHelper::set\_input\_handler ( [EntitySystem](#) & *ents*, tdt::uint *id*, const std::string & *handler* )

Changes the name of the table that contains a given entity's input handler.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>Name</i>	of the new input handler (Lua function).

### Note

The handler receives the ID of the entity and the key number when it's called, for current keybindings use the `game.enum.input` Lua table.

Definition at line 5 of file InputHelper.cpp.

## 5.19 level\_generators Namespace Reference

Namespace that contains different level generators that can be used in the game as well as the typedef for the DEFAULT\_LEVEL\_GENERATOR.

### Classes

- class [LevelGenerator](#)  
*Abstract parent class of all level generators, allows for different level generators used to create levels with minimal effort.*
- class [RandomLevelGenerator](#)  
*Level generator that uses simple RNG approach (counts the number of gold neighbours and increases the chance to spawn a gold deposit if needed).*

### Typedefs

- using [DEFAULT\\_LEVEL\\_GENERATOR](#) = [RandomLevelGenerator](#)  
*Typedef used as the default level generator when the game starts.*

#### 5.19.1 Detailed Description

Namespace that contains different level generators that can be used in the game as well as the typedef for the DEFAULT\_LEVEL\_GENERATOR.

#### 5.19.2 Typedef Documentation

##### 5.19.2.1 using level\_generators::DEFAULT\_LEVEL\_GENERATOR = typedef RandomLevelGenerator

Typedef used as the default level generator when the game starts.

Definition at line 83 of file LevelGenerators.hpp.

## 5.20 LightHelper Namespace Reference

Namespace containing auxiliary functions that help with the management of the light component.

## Functions

- void `set_visible` (`EntitySystem` &, `tdt::uint`, `bool`)  
*Sets the visibility status of a given light entity.*
- void `toggle_visible` (`EntitySystem` &, `tdt::uint`)  
*Toggles (visible -> invisible and vice versa) the visibility status of a given light entity.*
- bool `is_visible` (`EntitySystem` &, `tdt::uint`)  
*Returns the visibility status of a given light entity.*
- void `init` (`EntitySystem` &, `tdt::uint`)  
*Performs OGRE initialization of the light component of a given entity, used when loading a saved game.*

### 5.20.1 Detailed Description

Namespace containing auxiliary functions that help with the management of the light component.

### 5.20.2 Function Documentation

#### 5.20.2.1 void `LightHelper::init` ( `EntitySystem` & *ents*, `tdt::uint` *id* )

Performs OGRE initialization of the light component of a given entity, used when loading a saved game.

##### Parameters

<i>Entity</i>	system that contains the light entity.
<i>ID</i>	of the entity.

Definition at line 28 of file `LightHelper.cpp`.

#### 5.20.2.2 bool `LightHelper::is_visible` ( `EntitySystem` & *ents*, `tdt::uint` *id* )

Returns the visibility status of a given light entity.

##### Parameters

<i>Entity</i>	system that contains the light entity.
<i>ID</i>	of the entity.

Definition at line 19 of file `LightHelper.cpp`.

#### 5.20.2.3 void `LightHelper::set_visible` ( `EntitySystem` & *ents*, `tdt::uint` *id*, `bool` *val* )

Sets the visibility status of a given light entity.

##### Parameters

<i>Entity</i>	system that contains the light entity.
---------------	--

## Parameters

<i>ID</i>	of the entity.
<i>The</i>	new visibility status (true for visible).

Definition at line 5 of file LightHelper.cpp.

#### 5.20.2.4 void LightHelper::toggle\_visible ( EntitySystem & ents, tdt::uint id )

Toggles (visible -> invisible and vice versa) the visibility status of a given light entity.

## Parameters

<i>Entity</i>	system that contains the light entity.
<i>ID</i>	of the entity.

Definition at line 12 of file LightHelper.cpp.

## 5.21 LimitedLifeSpanHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the limited life span component.

### Functions

- void [set\\_max\\_time](#) (EntitySystem &, tdt::uint, tdt::real)  
*Sets the life span of a given entity.*
- tdt::real [get\\_max\\_time](#) (EntitySystem &, tdt::uint)  
*Returns the life span of a given entity.*
- tdt::real [get\\_curr\\_time](#) (EntitySystem &, tdt::uint)  
*Returns the life time of a given entity.*
- void [advance\\_curr\\_time](#) (EntitySystem &, tdt::uint, tdt::real)  
*Advances the life time of a given entity by a given value.*

#### 5.21.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the limited life span component.

#### 5.21.2 Function Documentation

##### 5.21.2.1 void LimitedLifeSpanHelper::advance\_curr\_time ( EntitySystem & ents, tdt::uint id, tdt::real val )

Advances the life time of a given entity by a given value.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	value to advance by.

Definition at line 30 of file LimitedLifeSpanHelper.cpp.

5.21.2.2 `tdt::real LimitedLifeSpanHelper::get_curr_time ( EntitySystem & ents, tdt::uint id )`

Returns the life time of a given entity.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 21 of file LimitedLifeSpanHelper.cpp.

5.21.2.3 `tdt::real LimitedLifeSpanHelper::get_max_time ( EntitySystem & ents, tdt::uint id )`

Returns the life span of a given entity.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file LimitedLifeSpanHelper.cpp.

5.21.2.4 `void LimitedLifeSpanHelper::set_max_time ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Sets the life span of a given entity.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new life span.

Definition at line 5 of file LimitedLifeSpanHelper.cpp.

## 5.22 ManaCrystalHelper Namespace Reference

Auxiliary namespace that contains functions that help with the management of the mana crystal component.



## Functions

- void [set\\_capacity](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the amount of max mana this entity adds to the player's mana capacity.*
- tdt::uint [get\\_capacity](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the amount of max mana this entity adds to the player's mana capacity.*
- void [set\\_regen](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the amount of mana regen this entity adds to the player's total mana regen.*
- tdt::uint [get\\_regen](#) ([EntitySystem](#) &, tdt::uint)  
*Sets the amount of mana regen this entity adds to the player's total mana regen.*

### 5.22.1 Detailed Description

Auxiliary namespace that contains functions that help with the management of the mana crystal component.

### 5.22.2 Function Documentation

#### 5.22.2.1 tdt::uint ManaCrystalHelper::get\_capacity ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the amount of max mana this entity adds to the player's mana capacity.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 19 of file ManaCrystalHelper.cpp.

#### 5.22.2.2 tdt::uint ManaCrystalHelper::get\_regen ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Sets the amount of mana regen this entity adds to the player's total mana regen.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 41 of file ManaCrystalHelper.cpp.

#### 5.22.2.3 void ManaCrystalHelper::set\_capacity ( [EntitySystem](#) & *ents*, tdt::uint *id*, tdt::uint *val* )

Sets the amount of max mana this entity adds to the player's mana capacity.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
------------------------------	---------------------------

## Parameters

<i>ID</i>	of the entity.
<i>The</i>	new capacity value.

Definition at line 6 of file ManaCrystalHelper.cpp.

#### 5.22.2.4 void ManaCrystalHelper::set\_regen ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the amount of mana regen this entity adds to the player's total mana regen.

## Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new capacity value.

Definition at line 28 of file ManaCrystalHelper.cpp.

## 5.23 ManaHelper Namespace Reference

Auxiliary namespace that contains functions that help with the management of the mana component.

### Functions

- void [add\\_mana](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Adds a given amount of mana to a given entity's current mana pool.*
- bool [sub\\_mana](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Removes a given amount of mana from a given entity's current mana pool.*
- void [set\\_mana](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the amount of mana a given entity has to a given amount.*
- tdt::uint [get\\_mana](#) (EntitySystem &, tdt::uint)  
*Returns the amount of mana a given entity has.*
- void [set\\_max\\_mana](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the max amount of mana a given entity can have.*
- tdt::uint [get\\_max\\_mana](#) (EntitySystem &, tdt::uint)  
*Returns the max amount of mana a given entity can have.*
- void [set\\_regen](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the mana regeneration value of a given entity.*
- tdt::uint [get\\_regen](#) (EntitySystem &, tdt::uint)  
*Returns the mana regeneration value of a given entity.*

#### 5.23.1 Detailed Description

Auxiliary namespace that contains functions that help with the management of the mana component.

### 5.23.2 Function Documentation

#### 5.23.2.1 void ManaHelper::add\_mana ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Adds a given amount of mana to a given entity's current mana pool.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>Amount</i>	of mana to add.

Definition at line 6 of file ManaHelper.cpp.

#### 5.23.2.2 tdt::uint ManaHelper::get\_mana ( EntitySystem & ents, tdt::uint id )

Returns the amount of mana a given entity has.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 48 of file ManaHelper.cpp.

#### 5.23.2.3 tdt::uint ManaHelper::get\_max\_mana ( EntitySystem & ents, tdt::uint id )

Returns the max amount of mana a given entity can have.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 72 of file ManaHelper.cpp.

#### 5.23.2.4 tdt::uint ManaHelper::get\_regen ( EntitySystem & ents, tdt::uint id )

Returns the mana regeneration value of a given entity.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 94 of file ManaHelper.cpp.

5.23.2.5 `void ManaHelper::set_mana ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the amount of mana a given entity has to a given amount.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new amount of mana.

Definition at line 35 of file ManaHelper.cpp.

5.23.2.6 `void ManaHelper::set_max_mana ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the max amount of mana a given entity can have.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new mana limit.

Definition at line 57 of file ManaHelper.cpp.

5.23.2.7 `void ManaHelper::set_regen ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the mana regeneration value of a given entity.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new mana regeneration value.

Definition at line 81 of file ManaHelper.cpp.

5.23.2.8 `bool ManaHelper::sub_mana ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Removes a given amount of mana from a given entity's current mana pool.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>Amount</i>	of mana to subtract.

Definition at line 19 of file ManaHelper.cpp.

## 5.24 MovementHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the movement component.

### Functions

- `tdt::real get_speed_modifier (EntitySystem &, tdt::uint)`  
*Returns the speed modifier of a given entity.*
- `void set_speed_modifier (EntitySystem &, tdt::uint, tdt::real)`  
*Changes the speed modifier of a given entity to a given value.*
- `Ogre::Vector3 dir_to (EntitySystem &, tdt::uint, tdt::uint)`  
*Returns the direction from a given entity to another given entity.*
- `Ogre::Vector3 get_dir (EntitySystem &, tdt::uint)`  
*Returns the direction a given entity is facing.*
- `Ogre::Vector3 get_dir_back (EntitySystem &, tdt::uint)`  
*Returns the opposite direction to the direction a given entity is facing.*
- `Ogre::Vector3 get_dir_left (EntitySystem &, tdt::uint)`  
*Returns the direction perpendicular to the direction a given entity is facing.*
- `Ogre::Vector3 get_dir_right (EntitySystem &, tdt::uint)`  
*Returns the direction perpendicular to the direction a given entity is facing.*
- `void set_original_speed (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the original value of a given entity's speed.*
- `tdt::real get_original_speed (EntitySystem &, tdt::uint)`  
*Returns the original speed of a given entity.*
- `void reset_speed (EntitySystem &, tdt::uint)`  
*Sets the speed of a given entity to it's original value.*

### 5.24.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the movement component.

### 5.24.2 Function Documentation

#### 5.24.2.1 `Ogre::Vector3 MovementHelper::dir_to ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )`

Returns the direction from a given entity to another given entity.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity.

Definition at line 21 of file MovementHelper.cpp.

#### 5.24.2.2 `Ogre::Vector3 MovementHelper::get_dir ( EntitySystem & ents, tdt::uint id )`

Returns the direction a given entity is facing.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 34 of file MovementHelper.cpp.

#### 5.24.2.3 `Ogre::Vector3 MovementHelper::get_dir_back ( EntitySystem & ents, tdt::uint id )`

Returns the opposite direction to the direction a given entity is facing.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 43 of file MovementHelper.cpp.

#### 5.24.2.4 `Ogre::Vector3 MovementHelper::get_dir_left ( EntitySystem & ents, tdt::uint id )`

Returns the direction perpendicular to the direction a given entity is facing.

(To the left.)

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 52 of file MovementHelper.cpp.

#### 5.24.2.5 `Ogre::Vector3 MovementHelper::get_dir_right ( EntitySystem & ents, tdt::uint id )`

Returns the direction perpendicular to the direction a given entity is facing.

(To the right.)

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 61 of file MovementHelper.cpp.

#### 5.24.2.6 `tdt::real MovementHelper::get_original_speed ( EntitySystem & ents, tdt::uint id )`

Returns the original speed of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 77 of file MovementHelper.cpp.

#### 5.24.2.7 `tdt::real MovementHelper::get_speed_modifier ( EntitySystem & ents, tdt::uint id )`

Returns the speed modifier of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 5 of file MovementHelper.cpp.

#### 5.24.2.8 `void MovementHelper::reset_speed ( EntitySystem & ents, tdt::uint id )`

Sets the speed of a given entity to it's original value.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 86 of file MovementHelper.cpp.

#### 5.24.2.9 `void MovementHelper::set_original_speed ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Sets the original value of a given entity's speed.

(Used for serialization.)

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 70 of file MovementHelper.cpp.

**5.24.2.10** void MovementHelper::set\_speed\_modifier ( EntitySystem & ents, tdt::uint id, tdt::real val )

Changes the speed modifier of a given entity to a given value.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>New</i>	speed value.

Definition at line 14 of file MovementHelper.cpp.

## 5.25 NameHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the name component.

### Functions

- void [set\\_name](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the name of a given entity.*
- const std::string & [get\\_name](#) (EntitySystem &, tdt::uint)  
*Returns the name of a given entity.*

### 5.25.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the name component.

### 5.25.2 Function Documentation

**5.25.2.1** const std::string & NameHelper::get\_name ( EntitySystem & ents, tdt::uint id )

Returns the name of a given entity.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file NameHelper.cpp.



### 5.25.2.2 void NameHelper::set\_name ( EntitySystem & ents, tdt::uint id, const std::string & val )

Sets the name of a given entity.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new name.

Definition at line 5 of file NameHelper.cpp.

## 5.26 NotificationHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the notification component.

### Functions

- void [set\\_cooldown](#) (EntitySystem &, tdt::uint, tdt::real)  
*Sets the cooldown between notifications a given entity can send to the game's log.*
- tdt::real [get\\_cooldown](#) (EntitySystem &, tdt::uint)  
*Returns the cooldown between notifications a given entity can send to the game's log.*
- void [reset](#) (EntitySystem &, tdt::uint)  
*Resets the cooldown between notifications a given entity can send to the game's log.*
- bool [notify](#) (EntitySystem &, tdt::uint, const std::string &)
- tdt::real [get\\_curr\\_time](#) (EntitySystem &, tdt::uint)  
*Returns the time since the last notification a given entity has sent to the game's log.*
- void [advance\\_curr\\_time](#) (EntitySystem &, tdt::uint, tdt::real)  
*Advances the time since the last notification a given entity has sent to the game's log.*

### 5.26.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the notification component.

### 5.26.2 Function Documentation

#### 5.26.2.1 void NotificationHelper::advance\_curr\_time ( EntitySystem & ents, tdt::uint id, tdt::real val )

Advances the time since the last notification a given entity has sent to the game's log.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>Time</i>	amount to advance by.

Definition at line 51 of file NotificationHelper.cpp.

#### 5.26.2.2 tdt::real NotificationHelper::get\_cooldown ( EntitySystem & ents, tdt::uint id )

Returns the cooldown between notifications a given entity can send to the game's log.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 13 of file NotificationHelper.cpp.

#### 5.26.2.3 tdt::real NotificationHelper::get\_curr\_time ( EntitySystem & ents, tdt::uint id )

Returns the time since the last notification a given entity has sent to the game's log.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 42 of file NotificationHelper.cpp.

#### 5.26.2.4 bool NotificationHelper::notify ( EntitySystem & ents, tdt::uint id, const std::string & msg )

##### Parameters

<i>Sends</i>	a notification to the game's log while respecting the notification cooldown of the entity that sends the notification. (And sets the cooldown if needed.)
<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 29 of file NotificationHelper.cpp.

#### 5.26.2.5 void NotificationHelper::reset ( EntitySystem & ents, tdt::uint id )

Resets the cooldown between notifications a given entity can send to the game's log.

(That means that the entity can notify immediately.)

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 22 of file NotificationHelper.cpp.

5.26.2.6 void NotificationHelper::set\_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the cooldown between notifications a given entity can send to the game's log.

#### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new cooldown.

Definition at line 6 of file NotificationHelper.cpp.

## 5.27 OnHitHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the on hit component.

### Functions

- void [set\\_blueprint](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the blueprint table handling incoming hits of a given entity.*
- const std::string & [get\\_blueprint](#) (EntitySystem &, tdt::uint)  
*Returns the name of the on hit blueprint of a given entity.*
- void [call](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Calls the on hit handler of a given entity.*
- void [set\\_cooldown](#) (EntitySystem &, tdt::uint, tdt::real)  
*Sets the cooldown between on hit blueprint calls of a given entity.*
- tdt::real [get\\_cooldown](#) (EntitySystem &, tdt::uint)  
*Returns the cooldown between on hit blueprint calls of a given entity.*

### 5.27.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the on hit component.

### 5.27.2 Function Documentation

5.27.2.1 void OnHitHelper::call ( EntitySystem & ents, tdt::uint id, tdt::uint hitter )

Calls the on hit handler of a given entity.

#### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>ID</i>	of the hitter. (Source of the attack.)

Definition at line 24 of file OnHitHelper.cpp.

5.27.2.2 `const std::string & OnHitHelper::get_blueprint ( EntitySystem & ents, tdt::uint id )`

Returns the name of the on hit blueprint of a given entity.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 13 of file OnHitHelper.cpp.

5.27.2.3 `tdt::real OnHitHelper::get_cooldown ( EntitySystem & ents, tdt::uint id )`

Returns the cooldown between on hit blueprint calls of a given entity.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 41 of file OnHitHelper.cpp.

5.27.2.4 `void OnHitHelper::set_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )`

Sets the blueprint table handling incoming hits of a given entity.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new blueprint name.

Definition at line 6 of file OnHitHelper.cpp.

5.27.2.5 `void OnHitHelper::set_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Sets the cooldown between on hit blueprint calls of a given entity.

#### Parameters

<a href="#"><i>EntitySystem</i></a>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new cooldown.

Definition at line 34 of file OnHitHelper.cpp.

## 5.28 PathfindingHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the pathfinding component.

### Functions

- const std::string & [get\\_pathfinding\\_blueprint](#) (EntitySystem &, tdt::uint)  
*Returns constant reference to the pathfinding blueprint of a given entity.*
- void [set\\_pathfinding\\_blueprint](#) (EntitySystem &, tdt::uint, const std::string &)  
*Changes the pathfinding blueprint of a given entity.*
- std::deque< tdt::uint > & [get\\_path](#) (EntitySystem &, tdt::uint)  
*Returns the node queue of a given entity's path.*
- bool [can\\_break](#) (tdt::uint, const PathfindingComponent &, tdt::uint)  
*Returns true if a given entity can break a structure residing on a given node (if any).*
- tdt::real [get\\_cost](#) (tdt::uint, const PathfindingComponent &, tdt::uint, DIRECTION::VAL)  
*Returns the cost a journey to a given node takes for a given entity.*

### 5.28.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the pathfinding component.

### 5.28.2 Function Documentation

#### 5.28.2.1 bool PathfindingHelper::can\_break ( tdt::uint *id1*, const PathfindingComponent & *comp*, tdt::uint *id2* )

Returns true if a given entity can break a structure residing on a given node (if any).

##### Parameters

<i>ID</i>	of the entity.
<i>Pathfinding</i>	component of the entity.
<i>ID</i>	of the node.

Definition at line 35 of file PathfindingHelper.cpp.

#### 5.28.2.2 tdt::real PathfindingHelper::get\_cost ( tdt::uint *id1*, const PathfindingComponent & *comp*, tdt::uint *id2*, DIRECTION::VAL *dir* )

Returns the cost a journey to a given node takes for a given entity.

## Parameters

<i>ID</i>	of the entity.
<i>Pathfinding</i>	component of the entity.
<i>ID</i>	of the node.

Definition at line 40 of file PathfindingHelper.cpp.

5.28.2.3 `std::deque< tdt::uint > & PathfindingHelper::get_path ( EntitySystem & ents, tdt::uint id )`

Returns the node queue of a given entity's path.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 24 of file PathfindingHelper.cpp.

5.28.2.4 `const std::string & PathfindingHelper::get_pathfinding_blueprint ( EntitySystem & ents, tdt::uint id )`

Returns constant reference to the pathfinding blueprint of a given entity.

(Which is used for the can\_pass & can\_break methods in lua.)

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the entity.
<i>ID</i>	of the entity.

Definition at line 6 of file PathfindingHelper.cpp.

5.28.2.5 `void PathfindingHelper::set_pathfinding_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & blueprint )`

Changes the pathfinding blueprint of a given entity.

## Parameters

<a href="#"><i>EntitySystem</i></a>	containing the entity.
<i>ID</i>	of the entity.
<i>Name</i>	of the new blueprint table.

Definition at line 17 of file PathfindingHelper.cpp.

## 5.29 PhysicsHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the physics component.

### Functions

- void [set\\_solid](#) ([EntitySystem](#) &, tdt::uint, bool)  
*Changes the solid state of a given entity.*
- bool [is\\_solid](#) ([EntitySystem](#) &, tdt::uint)  
*Returns true if a given entity is solid, false otherwise.*
- void [set\\_position](#) ([EntitySystem](#) &, tdt::uint, const Ogre::Vector3 &)  
*Sets the position of a given entity.*
- const Ogre::Vector3 & [get\\_position](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the position of a given entity.*
- void [set\\_half\\_height](#) ([EntitySystem](#) &, tdt::uint, tdt::real)  
*Sets the half height of a given entity (which is used to balance the fact that some models do not have their center on the level of their feet).*
- tdt::real [get\\_half\\_height](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the half height of a given entity (which is used to balance the fact that some models do not have their center on the level of their feet).*
- void [move\\_to](#) ([EntitySystem](#) &, tdt::uint, Ogre::Vector3)  
*Moves a given entity to a given point in space (absolute movement).*
- tdt::real [get\\_distance](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Returns the distance between two given entities.*
- tdt::real [get\\_angle](#) (Ogre::Vector3, Ogre::Vector3)  
*Returns the size of the angle bewteen two given vectors as tdt::real which can then be used in Ogre::Radian constructor for conversion to Radians.*
- void [set\\_2d\\_position](#) ([EntitySystem](#) &, tdt::uint, Ogre::Vector2)  
*Sets the position of a given entity disregarding it's Y coordinate.*
- Ogre::Vector2 [get\\_2d\\_position](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the X and Z coordinates of a given entity's position.*

### 5.29.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the physics component.

### 5.29.2 Function Documentation

#### 5.29.2.1 Ogre::Vector2 PhysicsHelper::get\_2d\_position ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the X and Z coordinates of a given entity's position.

#### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

**Note**

Cannot use const reference like in the usual `get_position` function because this method creates a proxy `Vector2` which cannot be passed as an lvalue reference due to it being a temporary object.

Definition at line 108 of file `PhysicsHelper.cpp`.

#### 5.29.2.2 `tdt::real PhysicsHelper::get_angle ( Ogre::Vector3 v1, Ogre::Vector3 v2 )`

Returns the size of the angle bewteen two given vectors as `tdt::real` which can then be used in `Ogre::Radian` constructor for conversion to Radians.

(The reason for this is that Lua does not have the notion of radians and as such using floating point numbers is easier.)

**Parameters**

<i>Vector</i>	#1.
<i>Vector</i>	#2.

Definition at line 88 of file `PhysicsHelper.cpp`.

#### 5.29.2.3 `tdt::real PhysicsHelper::get_distance ( EntitySystem & ents, tdt::uint id1, tdt::uint id2 )`

Returns the distance between two given entities.

**Parameters**

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity.

Definition at line 73 of file `PhysicsHelper.cpp`.

#### 5.29.2.4 `tdt::real PhysicsHelper::get_half_height ( EntitySystem & ents, tdt::uint id )`

Returns the half height of a given entity (which is used to balance the fact that some models do not have their center on the level of their feet).

**Parameters**

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 51 of file `PhysicsHelper.cpp`.



#### 5.29.2.5 `const Ogre::Vector3 & PhysicsHelper::get_position ( EntitySystem & ents, tdt::uint id )`

Returns the position of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components. Parma: ID of the entity.
------------------	--

Definition at line 34 of file PhysicsHelper.cpp.

#### 5.29.2.6 `bool PhysicsHelper::is_solid ( EntitySystem & ents, tdt::uint id )`

Returns true if a given entity is solid, false otherwise.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 13 of file PhysicsHelper.cpp.

#### 5.29.2.7 `void PhysicsHelper::move_to ( EntitySystem & ents, tdt::uint id, Ogre::Vector3 pos )`

Moves a given entity to a given point in space (absolute movement).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Target</i>	coordinate.

##### Note

The difference with `set_position` is that this function also sets the position of the scene node this entity is attached to (if such node exists) and thus moves the entity's model as well.

Definition at line 60 of file PhysicsHelper.cpp.

#### 5.29.2.8 `void PhysicsHelper::set_2d_position ( EntitySystem & ents, tdt::uint id, Ogre::Vector2 val )`

Sets the position of a given entity disregarding it's Y coordinate.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new position (vector2 containing the x and z coordinates.)

Definition at line 93 of file PhysicsHelper.cpp.

#### 5.29.2.9 void PhysicsHelper::set\_half\_height ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the half height of a given entity (which is used to balance the fact that some models do not have their center on the level of their feet).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new half height value.

Definition at line 44 of file PhysicsHelper.cpp.

#### 5.29.2.10 void PhysicsHelper::set\_position ( EntitySystem & ents, tdt::uint id, const Ogre::Vector3 & val )

Sets the position of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>The</i>	new position.

##### Note

The difference with move\_to is that this function does not set the position of the scene node this entity is attached to and thus does not move the entity's model as well.

Definition at line 22 of file PhysicsHelper.cpp.

#### 5.29.2.11 void PhysicsHelper::set\_solid ( EntitySystem & ents, tdt::uint id, bool val )

Changes the solid state of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>True</i>	for solid, false for non-solid.

Definition at line 6 of file PhysicsHelper.cpp.

## 5.30 PriceHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the price component.

### Functions

- void `set_price` (`EntitySystem` &, `tdt::uint`, `tdt::uint`)  
*Sets the price of a given entity.*
- `tdt::uint` `get_price` (`EntitySystem` &, `tdt::uint`)  
*Returns the price of a given entity.*

### 5.30.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the price component.

### 5.30.2 Function Documentation

#### 5.30.2.1 `tdt::uint PriceHelper::get_price ( EntitySystem & ents, tdt::uint id )`

Returns the price of a given entity.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.

Definition at line 12 of file PriceHelper.cpp.

#### 5.30.2.2 `void PriceHelper::set_price ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Sets the price of a given entity.

##### Parameters

<i>Entity</i>	system that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new price.

Definition at line 5 of file PriceHelper.cpp.

## 5.31 ProductHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the product component.

## Functions

- void [set\\_producer](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Set's the producer of a given entity (the building that spawned it).*
- tdt::uint [get\\_producer](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the producer of a given entity.*

### 5.31.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the product component.

### 5.31.2 Function Documentation

#### 5.31.2.1 tdt::uint ProductHelper::get\_producer ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Returns the producer of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 12 of file ProductHelper.cpp.

#### 5.31.2.2 void ProductHelper::set\_producer ( [EntitySystem](#) & *ents*, tdt::uint *id*, tdt::uint *producer* )

Set's the producer of a given entity (the building that spawned it).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>ID</i>	of the producer.

Definition at line 5 of file ProductHelper.cpp.

## 5.32 ProductionHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the production component.

## Functions

- void [set\\_production\\_blueprint](#) ([EntitySystem](#) &, tdt::uint, const std::string &)

- Changes the name of the blueprint table used to spawn new entities.*
- `const std::string & get_production_blueprint (EntitySystem &, tdt::uint)`  
*Returns the name of the blueprint table used to spawn new entities.*
- `void set_production_limit (EntitySystem &, tdt::uint, tdt::uint)`  
*Sets the maximal number of entities a given building can spawn.*
- `tdt::uint get_production_limit (EntitySystem &, tdt::uint)`  
*Returns the maximal number of entities a given building can spawn.*
- `void set_production_cooldown (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the time it takes for a given building to spawn a single entity.*
- `tdt::real get_production_cooldown (EntitySystem &, tdt::uint)`  
*Returns the time it takes for a given building to spawn a single entity.*
- `void set_production_progress (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the current spawning progress (in seconds, not %).*
- `tdt::real get_production_progress (EntitySystem &, tdt::uint)`  
*Returns the current spawning progress (in seconds, not %).*
- `void set_production_count (EntitySystem &, tdt::uint, tdt::uint)`  
*Sets the amount of entities spawned by a given building, but does not spawn or delete entities to match this number(!).*
- `tdt::uint get_production_count (EntitySystem &, tdt::uint)`  
*Returns the amount of entities spawned by a given building that are still alive.*

### 5.32.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the production component.

### 5.32.2 Function Documentation

#### 5.32.2.1 `const std::string & ProductionHelper::get_production_blueprint ( EntitySystem & ents, tdt::uint id )`

Returns the name of the blueprint table used to spawn new entities.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.

Definition at line 12 of file ProductionHelper.cpp.

#### 5.32.2.2 `tdt::real ProductionHelper::get_production_cooldown ( EntitySystem & ents, tdt::uint id )`

Returns the time it takes for a given building to spawn a single entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 46 of file ProductionHelper.cpp.

#### 5.32.2.3 `tdt::uint ProductionHelper::get_production_count ( EntitySystem & ents, tdt::uint id )`

Returns the amount of entities spawned by a given building that are still alive.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.

Definition at line 83 of file ProductionHelper.cpp.

#### 5.32.2.4 `tdt::uint ProductionHelper::get_production_limit ( EntitySystem & ents, tdt::uint id )`

Returns the maximal number of entities a given building can spawn.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.

Definition at line 30 of file ProductionHelper.cpp.

#### 5.32.2.5 `tdt::real ProductionHelper::get_production_progress ( EntitySystem & ents, tdt::uint id )`

Returns the current spawning progress (in seconds, not %).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.

Definition at line 67 of file ProductionHelper.cpp.

#### 5.32.2.6 `void ProductionHelper::set_production_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & blueprint )`

Changes the name of the blueprint table used to spawn new entities.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.
<i>Name</i>	of the new building table.

Definition at line 5 of file ProductionHelper.cpp.

#### 5.32.2.7 void ProductionHelper::set\_production\_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real cd )

Sets the time it takes for a given building to spawn a single entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.
<i>The</i>	new entity spawning time.

Definition at line 39 of file ProductionHelper.cpp.

#### 5.32.2.8 void ProductionHelper::set\_production\_count ( EntitySystem & ents, tdt::uint id, tdt::uint count )

Sets the amount of entities spawned by a given building, but does not spawn or delete entities to match this number(!).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.
<i>The</i>	new entity spawned amount.

Definition at line 76 of file ProductionHelper.cpp.

#### 5.32.2.9 void ProductionHelper::set\_production\_limit ( EntitySystem & ents, tdt::uint id, tdt::uint limit )

Sets the maximal number of entities a given building can spawn.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.
<i>The</i>	new entity limit.

Definition at line 23 of file ProductionHelper.cpp.

#### 5.32.2.10 void ProductionHelper::set\_production\_progress ( EntitySystem & ents, tdt::uint id, tdt::real prog )

Sets the current spawning progress (in seconds, not %).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the building.
<i>The</i>	new spawning progress time.

## Note

Time amounts above the cooldown will be adjusted to match the cooldown which will result into instant spawn.

Definition at line 55 of file ProductionHelper.cpp.

## 5.33 SpellHelper Namespace Reference

Auxiliary namespace that contains functions that help with the management of the spell component.

## Functions

- void `set_blueprint` (`EntitySystem` &, `tdt::uint`, `const std::string` &)  
*Sets the blueprint table of the spell a given entity can cast.*
- `const std::string` & `get_blueprint` (`EntitySystem` &, `tdt::uint`)  
*Returns the blueprint table of the spell a given entity can cast.*
- void `set_cooldown` (`EntitySystem` &, `tdt::uint`, `tdt::real`)  
*Sets the time period between casts of a given entity.*
- `tdt::real` `get_cooldown` (`EntitySystem` &, `tdt::uint`)  
*Returns the time period between casts of a given entity.*
- void `advance_curr_time` (`EntitySystem` &, `tdt::uint`, `tdt::real`)  
*Advances the timer before the next spell can be cast by a given entity.*
- void `set_curr_time` (`EntitySystem` &, `tdt::uint`, `tdt::real`)  
*Sets the timer before the next spell can be cast by a given entity.*
- `tdt::real` `get_curr_time` (`EntitySystem` &, `tdt::uint`)  
*Returns the cooldown timer value of a given entity.*
- void `cast` (`EntitySystem` &, `tdt::uint`)  
*Makes a given entity to cast it's spell (if possible).*

### 5.33.1 Detailed Description

Auxiliary namespace that contains functions that help with the management of the spell component.

### 5.33.2 Function Documentation

#### 5.33.2.1 void SpellHelper::advance\_curr\_time ( EntitySystem & ents, tdt::uint id, tdt::real val )

Advances the timer before the next spell can be cast by a given entity.



## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>Time</i>	to advance by.

Definition at line 38 of file SpellHelper.cpp.

#### 5.33.2.2 void SpellHelper::cast ( EntitySystem & ents, tdt::uint id )

Makes a given entity to cast it's spell (if possible).

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

## Note

Ignores cooldown.

Definition at line 61 of file SpellHelper.cpp.

#### 5.33.2.3 const std::string & SpellHelper::get\_blueprint ( EntitySystem & ents, tdt::uint id )

Returns the blueprint table of the spell a given entity can cast.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 13 of file SpellHelper.cpp.

#### 5.33.2.4 tdt::real SpellHelper::get\_cooldown ( EntitySystem & ents, tdt::uint id )

Returns the time period between casts of a given entity.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 29 of file SpellHelper.cpp.

#### 5.33.2.5 `tdt::real SpellHelper::get_curr_time ( EntitySystem & ents, tdt::uint id )`

Returns the cooldown timer value of a given entity.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 52 of file SpellHelper.cpp.

#### 5.33.2.6 `void SpellHelper::set_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )`

Sets the blueprint table of the spell a given entity can cast.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new spell blueprint.

Definition at line 6 of file SpellHelper.cpp.

#### 5.33.2.7 `void SpellHelper::set_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Sets the time period between casts of a given entity.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new cooldown value.

Definition at line 22 of file SpellHelper.cpp.

#### 5.33.2.8 `void SpellHelper::set_curr_time ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Sets the timer before the next spell can be cast by a given entity.

##### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	new timer value (cooldown - timer == time remaining).

Definition at line 45 of file SpellHelper.cpp.

## 5.34 StructureHelper Namespace Reference

Auxiliary namespace containing the functions that help with the management of the structure component.

### Functions

- void [add\\_residences](#) ([EntitySystem](#) &, tdt::uint, const std::vector< tdt::uint > &)  
*Adds the members of a given vector (containing node IDs) as residences of a given entity (that has a structure component).*
- void [add\\_residence](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Adds a single node as a residence to the residence list of a given entity.*
- void [set\\_radius](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the radius of the area a given structure occupies.*
- tdt::uint [get\\_radius](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the radius of a structure (amount of grid nodes from the centre of the structure to one of the sides - not including the centre).*
- void [set\\_walk\\_through](#) ([EntitySystem](#) &, tdt::uint, bool)  
*Sets the walk through field of a structure, causing it to either block or allow pathfinding.*
- bool [is\\_walk\\_through](#) ([EntitySystem](#) &, tdt::uint)  
*Returns true if pathfinding is possible through the structure, false otherwise.*

### 5.34.1 Detailed Description

Auxiliary namespace containing the functions that help with the management of the structure component.

### 5.34.2 Function Documentation

#### 5.34.2.1 void StructureHelper::add\_residence ( EntitySystem & ents, tdt::uint ent\_id, tdt::uint node\_id )

Adds a single node as a residence to the residence list of a given entity.

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the structure.
<i>ID</i>	of the residence.

Definition at line 15 of file StructureHelper.cpp.

#### 5.34.2.2 void StructureHelper::add\_residences ( EntitySystem & ents, tdt::uint ent\_id, const std::vector< tdt::uint > & residences )

Adds the members of a given vector (containing node IDs) as residences of a given entity (that has a structure component).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>Vector</i>	containing the IDs of the new residences (nodes the entity is on).

Definition at line 5 of file StructureHelper.cpp.

#### 5.34.2.3 `tdt::uint StructureHelper::get_radius ( EntitySystem & ents, tdt::uint id )`

Returns the radius of a structure (amount of grid nodes from the centre of the structure to one of the sides - not including the centre).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.

Definition at line 29 of file StructureHelper.cpp.

#### 5.34.2.4 `bool StructureHelper::is_walk_through ( EntitySystem & ents, tdt::uint id )`

Returns true if pathfinding is possible through the structure, false otherwise.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the structure.

Definition at line 45 of file StructureHelper.cpp.

#### 5.34.2.5 `void StructureHelper::set_radius ( EntitySystem & ents, tdt::uint id, tdt::uint radius )`

Sets the radius of the area a given structure occupies.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity (which has a structure component).
<i>The</i>	new radius.

Definition at line 22 of file StructureHelper.cpp.

#### 5.34.2.6 `void StructureHelper::set_walk_through ( EntitySystem & ents, tdt::uint id, bool on_off )`

Sets the walk through field of a structure, causing it to either block or allow pathfinding.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the structure.
<i>True</i>	for walkable, false for not walkable.

Definition at line 38 of file StructureHelper.cpp.

## 5.35 TaskHandlerHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the task handler component.

### Functions

- `std::deque< tdt::uint > & get_task_queue (EntitySystem &, tdt::uint)`  
*Returns a reference to the task queue of a given entity.*
- `bool task_possible (EntitySystem &, tdt::uint, tdt::uint)`  
*Checks whether an entity can accept and complete a given task.*
- `bool task_possible (EntitySystem &, tdt::uint, TASK_TYPE)`  
*Checks whether an entity can accept and complete a task of a given task type.*
- `void clear_task_queue (EntitySystem &, tdt::uint)`  
*Cancels all tasks in a given entity's task queue.*
- `void add_possible_task (EntitySystem &, tdt::uint, TASK_TYPE)`  
*Marks a given entity as available to accept tasks of a given task type.*
- `void delete_possible_task (EntitySystem &, tdt::uint, TASK_TYPE)`  
*Marks a given entity as unavailable to accept tasks of a given task type.*
- `void set_blueprint (EntitySystem &, tdt::uint, const std::string &)`  
*Sets the handling blueprint of a given entity.*
- `const std::string & get_blueprint (EntitySystem &, tdt::uint)`  
*Returns the handling blueprint of a given entity.*

### 5.35.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the task handler component.

### 5.35.2 Function Documentation

#### 5.35.2.1 void TaskHandlerHelper::add\_possible\_task ( EntitySystem & ents, tdt::uint id, TASK\_TYPE type )

Marks a given entity as available to accept tasks of a given task type.

## Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>Type</i>	of the task to be added.

Definition at line 51 of file TaskHandlerHelper.cpp.

#### 5.35.2.2 void TaskHandlerHelper::clear\_task\_queue ( EntitySystem & ents, tdt::uint id )

Cancels all tasks in a given entity's task queue.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 36 of file TaskHandlerHelper.cpp.

#### 5.35.2.3 void TaskHandlerHelper::delete\_possible\_task ( EntitySystem & ents, tdt::uint id, TASK\_TYPE type )

Marks a given entity as unavailable to accept tasks of a given task type.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.
<i>Type</i>	of the task to be deleted.

Definition at line 58 of file TaskHandlerHelper.cpp.

#### 5.35.2.4 const std::string & TaskHandlerHelper::get\_blueprint ( EntitySystem & ents, tdt::uint id )

Returns the handling blueprint of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 72 of file TaskHandlerHelper.cpp.

#### 5.35.2.5 std::deque< tdt::uint > & TaskHandlerHelper::get\_task\_queue ( EntitySystem & ents, tdt::uint id )

Returns a reference to the task queue of a given entity.

##### Parameters

<a href="#">EntitySystem</a>	that contains the entity.
<i>ID</i>	of the entity.

Definition at line 5 of file TaskHandlerHelper.cpp.

5.35.2.6 `void TaskHandlerHelper::set_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )`

Sets the handling blueprint of a given entity.

#### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>The</i>	name of the new blueprint.

Definition at line 65 of file TaskHandlerHelper.cpp.

5.35.2.7 `bool TaskHandlerHelper::task_possible ( EntitySystem & ents, tdt::uint ent_id, tdt::uint task_id )`

Checks whether an entity can accept and complete a given task.

#### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>ID</i>	of the task.

Definition at line 15 of file TaskHandlerHelper.cpp.

5.35.2.8 `bool TaskHandlerHelper::task_possible ( EntitySystem & ents, tdt::uint id, TASK_TYPE val )`

Checks whether an entity can accept and complete a task of agiven task type.

#### Parameters

<i>EntitySystem</i>	that contains the entity.
<i>ID</i>	of the entity.
<i>Task</i>	type to be tested.

Definition at line 27 of file TaskHandlerHelper.cpp.

## 5.36 TaskHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the task component.

## Functions

- void [set\\_task\\_source](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the source of a given task (that is, the entity that is completing the task).*
- tdt::uint [get\\_task\\_source](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the source of a given task (that is, the entity that is completing the task).*
- void [set\\_task\\_target](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Sets the target entity of a given task.*
- tdt::uint [get\\_task\\_target](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the target entity of a given task.*
- void [set\\_task\\_type](#) ([EntitySystem](#) &, tdt::uint, TASK\_TYPE)  
*Sets the task type of a given task.*
- TASK\_TYPE [get\\_task\\_type](#) ([EntitySystem](#) &, tdt::uint)  
*Returns the task type of a given task.*
- void [add\\_task](#) ([EntitySystem](#) &, tdt::uint, tdt::uint, bool=false)  
*Assigns a new task to an entity (by adding it to the task queue).*
- tdt::uint [create\\_task](#) ([EntitySystem](#) &, tdt::uint, TASK\_TYPE)  
*Creates a new task of a given tasks and returns it's ID.*
- void [cancel\\_task](#) ([EntitySystem](#) &, tdt::uint)  
*Destroys the [TaskComponent](#) of a given task, effectively stopping it's completion.*
- void [set\\_complete](#) ([EntitySystem](#) &, tdt::uint)  
*Sets a given task to a complete state.*
- bool [is\\_complete](#) ([EntitySystem](#) &, tdt::uint)  
*Returns true if a given task is complete, false otherwise.*

### 5.36.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the task component.

### 5.36.2 Function Documentation

#### 5.36.2.1 void TaskHelper::add\_task ( [EntitySystem](#) & *ents*, tdt::uint *ent\_id*, tdt::uint *task\_id*, bool *priority = false* )

Assigns a new task to an entity (by adding it to the task queue).

##### Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the entity.
<i>ID</i>	of the task.
<i>If</i>	true, the task will be added to the fron of the task queue.

Definition at line 53 of file TaskHelper.cpp.

#### 5.36.2.2 void TaskHelper::cancel\_task ( [EntitySystem](#) & *ents*, tdt::uint *task\_id* )

Destroys the [TaskComponent](#) of a given task, effectively stopping it's completion.



## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the task.

Definition at line 95 of file TaskHelper.cpp.

### 5.36.2.3 tdt::uint TaskHelper::create\_task ( EntitySystem & ents, tdt::uint target, TASK\_TYPE type )

Creates a new task of a given tasks and returns it's ID.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the task's target (goto location, kill target etc.).
<i>Type</i>	of the task.

Definition at line 80 of file TaskHelper.cpp.

### 5.36.2.4 tdt::uint TaskHelper::get\_task\_source ( EntitySystem & ents, tdt::uint id )

Returns the source of a given task (that is, the entity that is completing the task).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.

Definition at line 12 of file TaskHelper.cpp.

### 5.36.2.5 tdt::uint TaskHelper::get\_task\_target ( EntitySystem & ents, tdt::uint id )

Returns the target entity of a given task.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.

Definition at line 28 of file TaskHelper.cpp.

### 5.36.2.6 TASK\_TYPE TaskHelper::get\_task\_type ( EntitySystem & ents, tdt::uint id )

Returns the task type of a given task.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.

Definition at line 44 of file TaskHelper.cpp.

**5.36.2.7 bool TaskHelper::is\_complete ( EntitySystem & ents, tdt::uint id )**

Returns true if a given task is complete, false otherwise.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the task.

Definition at line 122 of file TaskHelper.cpp.

**5.36.2.8 void TaskHelper::set\_complete ( EntitySystem & ents, tdt::uint id )**

Sets a given task to a complete state.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>ID</i>	of the task.

Definition at line 115 of file TaskHelper.cpp.

**5.36.2.9 void TaskHelper::set\_task\_source ( EntitySystem & ents, tdt::uint id, tdt::uint source )**

Sets the source of a given task (that is, the entity that is completing the task).

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.
<i>Source</i>	ID.

Definition at line 5 of file TaskHelper.cpp.

**5.36.2.10 void TaskHelper::set\_task\_target ( EntitySystem & ents, tdt::uint id, tdt::uint target )**

Sets the target entity of a given task.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.
<i>Target</i>	ID.

Definition at line 21 of file TaskHelper.cpp.

5.36.2.11 void TaskHelper::set\_task\_type ( EntitySystem & *ents*, tdt::uint *id*, TASK\_TYPE *type* )

Sets the task type of a given task.

## Parameters

<i>Reference</i>	to the entity system containing components.
<i>Task</i>	ID.
<i>The</i>	new task type.

Definition at line 37 of file TaskHelper.cpp.

## 5.37 tdt Namespace Reference

Namespace containing numeric types used in the game.

### Typedefs

- using **uint** = std::size\_t
- using **real** = Ogre::Real

#### 5.37.1 Detailed Description

Namespace containing numeric types used in the game.

## 5.38 TimeHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the time component.

## Functions

- `tdt::real get_curr_time (EntitySystem &, tdt::uint)`  
*Returns the time that has passed since the timer started.*
- `void advance_curr_time (EntitySystem &, tdt::uint, tdt::real)`  
*Adds a given time value to a given timer.*
- `void max_curr_time (EntitySystem &, tdt::uint)`  
*Completes a given timer (by maxing it's current time).*
- `void set_time_limit (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the time a given timer requires for completion.*
- `tdt::real get_time_limit (EntitySystem &, tdt::uint)`  
*Returns the time a given timer requires for completion.*
- `void set_target (EntitySystem &, tdt::uint, tdt::uint)`  
*Sets the ID of the event a given timer starts/ends.*
- `tdt::uint get_target (EntitySystem &, tdt::uint)`  
*Returns the ID of the event a given timer starts/ends.*
- `void set_type (EntitySystem &, tdt::uint, TIME_EVENT)`  
*Sets the type of a given timer.*
- `TIME_EVENT get_type (EntitySystem &, tdt::uint)`  
*Returns the type of a given timer.*

### 5.38.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the time component.

### 5.38.2 Function Documentation

#### 5.38.2.1 `void TimeHelper::advance_curr_time ( EntitySystem & ents, tdt::uint id, tdt::real val )`

Adds a given time value to a given timer.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.
<i>Time</i>	to add.

Definition at line 14 of file TimeHelper.cpp.

#### 5.38.2.2 `tdt::real TimeHelper::get_curr_time ( EntitySystem & ents, tdt::uint id )`

Returns the time that has passed since the timer started.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of th entity.

Definition at line 5 of file TimeHelper.cpp.

#### 5.38.2.3 tdt::uint TimeHelper::get\_target ( EntitySystem & ents, tdt::uint id )

Returns the ID of the event a given timer starts/ends.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.

Definition at line 51 of file TimeHelper.cpp.

#### 5.38.2.4 tdt::real TimeHelper::get\_time\_limit ( EntitySystem & ents, tdt::uint id )

Returns the time a given timer requires for completion.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.

Definition at line 35 of file TimeHelper.cpp.

#### 5.38.2.5 TIME\_EVENT TimeHelper::get\_type ( EntitySystem & ents, tdt::uint id )

Returns the type of a given timer.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.

Definition at line 67 of file TimeHelper.cpp.

#### 5.38.2.6 void TimeHelper::max\_curr\_time ( EntitySystem & ents, tdt::uint id )

Completes a given timer (by maxing it's current time).

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.

Definition at line 21 of file TimeHelper.cpp.

#### 5.38.2.7 void TimeHelper::set\_target ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the ID of the event a given timer starts/ends.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.
<i>ID</i>	of the event.

Definition at line 44 of file TimeHelper.cpp.

#### 5.38.2.8 void TimeHelper::set\_time\_limit ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the time a given timer requires for completion.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.
<i>The</i>	new time limit.

Definition at line 28 of file TimeHelper.cpp.

#### 5.38.2.9 void TimeHelper::set\_type ( EntitySystem & ents, tdt::uint id, TIME\_EVENT val )

Sets the type of a given timer.

##### Parameters

<i>Entity</i>	system which contains the timer.
<i>ID</i>	of the timer.
<i>The</i>	new type.

Definition at line 60 of file TimeHelper.cpp.

## 5.39 TriggerHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the trigger component.

### Functions

- void [set\\_blueprint](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the blueprint table used used to handle triggering of a given entity.*

- `const std::string & get_blueprint (EntitySystem &, tdt::uint)`  
*Returns the blueprint table used used to handle triggering of a given entity.*
- `void set_linked_entity (EntitySystem &, tdt::uint, tdt::uint)`
- `tdt::uint get_linked_entity (EntitySystem &, tdt::uint)`
- `void set_cooldown (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the cooldown before an entity can be triggered again.*
- `tdt::real get_cooldown (EntitySystem &, tdt::uint)`  
*Returns the cooldown before an entity can be triggered again.*
- `void trigger (EntitySystem &, tdt::uint, tdt::uint)`  
*Triggers an entity.*
- `bool can_be_triggered_by (EntitySystem &, tdt::uint, tdt::uint)`  
*Returns true if a given entity can be triggered by another given entity.*
- `void reset_timer (EntitySystem &, tdt::uint)`  
*Sets the trigger cooldown timer of a given entity to zero.*
- `void set_radius (EntitySystem &, tdt::uint, tdt::real)`  
*Sets the trigger radius of a given entity.*
- `tdt::real get_radius (EntitySystem &, tdt::uint)`  
*Returns the trigger radius of a given entity.*

### 5.39.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the trigger component.

### 5.39.2 Function Documentation

#### 5.39.2.1 `bool TriggerHelper::can_be_triggered_by ( EntitySystem & ents, tdt::uint id, tdt::uint target )`

Returns true if a given entity can be triggered by another given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity that is supposed to get triggered.
<i>ID</i>	of the triggering entity.

Definition at line 61 of file TriggerHelper.cpp.

#### 5.39.2.2 `const std::string & TriggerHelper::get_blueprint ( EntitySystem & ents, tdt::uint id )`

Returns the blueprint table used used to handle triggering of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 13 of file TriggerHelper.cpp.

#### 5.39.2.3 tdt::real TriggerHelper::get\_cooldown ( EntitySystem & ents, tdt::uint id )

Returns the cooldown before an entity can be triggered again.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 45 of file TriggerHelper.cpp.

#### 5.39.2.4 tdt::uint TriggerHelper::get\_linked\_entity ( EntitySystem & ents, tdt::uint id )

##### Parameters

<i>Returns</i>	the ID of the entity a given trigger entity is linked to.
<i>Entity</i>	system containing the entity.
<i>ID</i>	of the trigger entity.

Definition at line 29 of file TriggerHelper.cpp.

#### 5.39.2.5 tdt::real TriggerHelper::get\_radius ( EntitySystem & ents, tdt::uint id )

Returns the trigger radius of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 91 of file TriggerHelper.cpp.

#### 5.39.2.6 void TriggerHelper::reset\_timer ( EntitySystem & ents, tdt::uint id )

Sets the trigger cooldown timer of a given entity to zero.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 77 of file TriggerHelper.cpp.



#### 5.39.2.7 void TriggerHelper::set\_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )

Sets the blueprint table used used to handle triggering of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new blueprint name.

Definition at line 6 of file TriggerHelper.cpp.

#### 5.39.2.8 void TriggerHelper::set\_cooldown ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the cooldown before an entity can be triggered again.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 38 of file TriggerHelper.cpp.

#### 5.39.2.9 void TriggerHelper::set\_linked\_entity ( EntitySystem & ents, tdt::uint id, tdt::uint val )

##### Parameters

<i>Sets</i>	the linked entity a given trigger entity is linked to.
<i>Entity</i>	system containing the entity.
<i>ID</i>	of the trigger entity.
<i>ID</i>	of the linked entity.

Definition at line 22 of file TriggerHelper.cpp.

#### 5.39.2.10 void TriggerHelper::set\_radius ( EntitySystem & ents, tdt::uint id, tdt::real val )

Sets the trigger radius of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 84 of file TriggerHelper.cpp.

#### 5.39.2.11 void TriggerHelper::trigger ( EntitySystem & ents, tdt::uint id, tdt::uint target )

Triggers an entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the triggered entity.
<i>ID</i>	of the entity that triggered the triggered entity.

Definition at line 54 of file TriggerHelper.cpp.

## 5.40 UpgradeHelper Namespace Reference

Auxiliary namespace containing functions that help with the management of the upgrade component.

### Functions

- void [set\\_blueprint](#) (EntitySystem &, tdt::uint, const std::string &)  
*Sets the blueprint table that handles the upgrading of a given entity.*
- const std::string & [get\\_blueprint](#) (EntitySystem &, tdt::uint)  
*Returns the blueprint table that handles the upgrading of a given entity.*
- tdt::uint [set\\_experience](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the amount of experience a given entity has.*
- tdt::uint [get\\_experience](#) (EntitySystem &, tdt::uint)  
*Returns the amount of experience a given entity has.*
- tdt::uint [add\\_experience](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Adds a given amount of experience to a given entity.*
- void [set\\_exp\\_needed](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the amount of experience needed for next level of a given entity.*
- tdt::uint [get\\_exp\\_needed](#) (EntitySystem &, tdt::uint)  
*Returns the amount of experience needed for next level of a given entity.*
- void [set\\_level](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the level of a given entity.*
- tdt::uint [get\\_level](#) (EntitySystem &, tdt::uint)  
*Returns the level of a given entity.*
- void [set\\_level\\_cap](#) (EntitySystem &, tdt::uint, tdt::uint)  
*Sets the maximum level a given entity can reach.*
- tdt::uint [get\\_level\\_cap](#) (EntitySystem &, tdt::uint)  
*Returns the maximum level a given entity can reach.*
- bool [can\\_level\\_up](#) (EntitySystem &, tdt::uint)  
*Returns true if a given entity can level up.*
- void [upgrade](#) (EntitySystem &, tdt::uint)  
*Upgrades a given entity that can level up.*

#### 5.40.1 Detailed Description

Auxiliary namespace containing functions that help with the management of the upgrade component.

### 5.40.2 Function Documentation

#### 5.40.2.1 `tdt::uint UpgradeHelper::add_experience ( EntitySystem & ents, tdt::uint id, tdt::uint val )`

Adds a given amount of experience to a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	amount of experience to be added.

Definition at line 59 of file UpgradeHelper.cpp.

#### 5.40.2.2 `bool UpgradeHelper::can_level_up ( EntitySystem & ents, tdt::uint id )`

Returns true if a given entity can level up.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 159 of file UpgradeHelper.cpp.

#### 5.40.2.3 `const std::string & UpgradeHelper::get_blueprint ( EntitySystem & ents, tdt::uint id )`

Returns the blueprint table that handles the upgrading of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 14 of file UpgradeHelper.cpp.

#### 5.40.2.4 `tdt::uint UpgradeHelper::get_exp_needed ( EntitySystem & ents, tdt::uint id )`

Returns the amount of experience needed for next level of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 106 of file UpgradeHelper.cpp.

#### 5.40.2.5 `tdt::uint UpgradeHelper::get_experience ( EntitySystem & ents, tdt::uint id )`

Returns the amount of experience a given entity has.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 50 of file UpgradeHelper.cpp.

#### 5.40.2.6 `tdt::uint UpgradeHelper::get_level ( EntitySystem & ents, tdt::uint id )`

Returns the level of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 128 of file UpgradeHelper.cpp.

#### 5.40.2.7 `tdt::uint UpgradeHelper::get_level_cap ( EntitySystem & ents, tdt::uint id )`

Returns the maximum level a given entity can reach.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 150 of file UpgradeHelper.cpp.

#### 5.40.2.8 `void UpgradeHelper::set_blueprint ( EntitySystem & ents, tdt::uint id, const std::string & val )`

Sets the blueprint table that handles the upgrading of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new blueprint name.

Definition at line 7 of file UpgradeHelper.cpp.

#### 5.40.2.9 void UpgradeHelper::set\_exp\_needed ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the amount of experience needed for next level of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new experience amount needed.

Definition at line 86 of file UpgradeHelper.cpp.

#### 5.40.2.10 tdt::uint UpgradeHelper::set\_experience ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the amount of experience a given entity has.

Won't allow more experience than is needed for the next level and returns the remaining experience not added.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new experience amount.

Definition at line 23 of file UpgradeHelper.cpp.

#### 5.40.2.11 void UpgradeHelper::set\_level ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the level of a given entity.

##### Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new level.

##### Note

Does not change the attributes of the entity!

Definition at line 115 of file UpgradeHelper.cpp.

#### 5.40.2.12 void UpgradeHelper::set\_level\_cap ( EntitySystem & ents, tdt::uint id, tdt::uint val )

Sets the maximum level a given entity can reach.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.
<i>The</i>	new max level.

Definition at line 137 of file UpgradeHelper.cpp.

#### 5.40.2.13 void UpgradeHelper::upgrade ( EntitySystem & ents, tdt::uint id )

Upgrades a given entity that can level up.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 168 of file UpgradeHelper.cpp.

## 5.41 util Namespace Reference

The util namespace contains functors used as conditions in searches and other helper structures/functions used throughout the code.

### Namespaces

- [effect](#)  
*Contains effect functors that perform an action on the entity they are called on.*
- [heuristic](#)  
*Forward declarations.*
- [path\\_type](#)  
*Contains different path types, which are used to check if a path should be returned once found or when an augmenting edge is found to the path.*
- [pathfinding](#)  
*Contains the pathfinding algorithms used by the [util::pathfind](#) function.*

### Classes

- class [EntityDestroyer](#)  
*A structure providing the private method [EntitySystem::destroy\\_entity](#) to the [DestructorHelper::destroy](#) function.*
- struct [HAS\\_GOLD](#)  
*Tests if a given entity has a gold component.*
- struct [IS\\_ENEMY](#)  
*Tests if if the entity it is called on is an enemy of the entity specified in it's constructor.*
- struct [IS\\_FRIENDLY](#)  
*Tests if if the entity it is called on is a friend of the entity specified in it's constructor.*
- struct [IS\\_FRIENDLY\\_OR\\_NEUTRAL](#)  
*Tests if if the entity it is called on is a friend of or neutral to the entity specified in it's constructor.*
- struct [IS\\_GOLD\\_VAULT](#)  
*Tests if a given entity is of friendly faction, has structure component and has gold component (that is, it's a gold vault).*

## Typedefs

- using `DEFAULT_PATH_TYPE` = `path_type::FIRST_PATH`  
*Default types for the different pathfinding functors.*
- using `DEFAULT_HEURISTIC` = `heuristic::PORTAL_HEURISTIC`
- using `DEFAULT_PATHFINDING_ALGORITHM` = `pathfinding::A_STAR< DEFAULT_PATH_TYPE >`

## Functions

- `template<typename ALGORITHM = util::DEFAULT_PATHFINDING_ALGORITHM>`  
`bool pathfind (EntitySystem &ents, tdt::uint id, tdt::uint target, util::heuristic::HEURISTIC &heuristic, bool add_path=true, bool allow_destruction=true)`  
*Finds a path using a given algorithm (specified as a template parameter) and heuristic and adds the path to the pathfinding entity if needed.*
- `int get_enum_direction (EntitySystem &, tdt::uint, tdt::uint)`  
*Returns the direction from a given entity to another given entity in the form of the direction enum (8 directional).*
- `tdt::uint get_random (tdt::uint, tdt::uint)`  
*Returns a random number within a given range.*
- `tdt::uint abs (int)`  
*Returns the absolute value of a given integer.*

### 5.41.1 Detailed Description

The util namespace contains functors used as conditions in searches and other helper structures/functions used throughout the code.

The utim namespace contains various tools and utilities used by the game's engine.

Util namespace contains general tools and utilities used by the game's engine.

This part of the namespace contains functions used for pathfinding.

This part contains the pathfinding algorithms, heuristics and path types. Also contains the typedefs for `DEFAULT_PATH_TYPE`, `DEFAULT_PATHFINDING_ALGORITHM` and `DEFAULT_HEURISTIC`.

### 5.41.2 Typedef Documentation

#### 5.41.2.1 using util::DEFAULT\_PATH\_TYPE = typedef path\_type::FIRST\_PATH

Default types for the different pathfinding functors.

Definition at line 339 of file PathfindingAlgorithms.hpp.

### 5.41.3 Function Documentation

#### 5.41.3.1 tdt::uint util::abs ( int val )

Returns the absolute value of a given integer.

## Parameters

<i>The</i>	number we want absolute value of.
------------	-----------------------------------

Definition at line 102 of file Util.cpp.

#### 5.41.3.2 `int util::get_enum_direction ( EntitySystem & ents, tdt::uint id, tdt::uint target )`

Returns the direction from a given entity to another given entity in the form of the direction enum (8 directional).

## Parameters

<i>EntitySystem</i>	containing both entities.
<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity.

## Note

The direction is #1 -> #2.

Definition at line 54 of file Util.cpp.

#### 5.41.3.3 `tdt::uint util::get_random ( tdt::uint min, tdt::uint max )`

Returns a random number within a given range.

## Parameters

<i>Lower</i>	bound of the range.
<i>Upper</i>	bound of the range.

Definition at line 93 of file Util.cpp.

#### 5.41.3.4 `template<typename ALGORITHM = util::DEFAULT_PATHFINDING_ALGORITHM> bool util::pathfind ( EntitySystem & ents, tdt::uint id, tdt::uint target, util::heuristic::HEURISTIC & heuristic, bool add_path = true, bool allow_destruction = true )`

Finds a path using a given algorithm (specified as a template parameter) and heuristic and adds the path to the pathfinding entity if needed.

## Parameters

<i>Entity</i>	system containing the entity and the pathfinding grid.
<i>ID</i>	of the pathfinding entity.
<i>Target</i>	of the pathfinding.
<i>Heuristic</i>	used by the pathfinding algorithm.
<i>If</i>	true, the path will be added to the pathdinding entity's pathfinding component.
<i>If</i>	true, the entity will be allowed to destroy blocks on it's way.



Definition at line 26 of file Pathfinding.hpp.

## 5.42 util::effect Namespace Reference

Contains effect functors that perform an action on the entity they are called on.

### Classes

- struct [DAMAGE\\_EFFECT](#)  
*Deals a random damage in a given range to the entity it's called on.*
- struct [FREEZE\\_EFFECT](#)  
*Freezes a given entity in place for a given time period.*
- struct [HEAL\\_EFFECT](#)  
*Fully heals the entity it's called on.*
- struct [LOWER\\_SPEED\\_EFFECT](#)  
*Halves the speed of the entity it's called on for a given time period.*

### 5.42.1 Detailed Description

Contains effect functors that perform an action on the entity they are called on.

## 5.43 util::heuristic Namespace Reference

Forward declarations.

### Classes

- struct [HEURISTIC](#)  
*Abstract parent of all heuristics.*
- struct [MANHATTAN\\_DISTANCE](#)  
*Returns the manhattan distance between two nodes.*
- struct [NO\\_HEURISTIC](#)  
*Represents no heuristic by returning 0 all the time.*
- struct [PORTAL\\_HEURISTIC](#)  
*Variation of the Manhattan distance heuristic that takes portals into accounts.*
- struct [RUN\\_AWAY\\_HEURISTIC](#)  
*Used by entities that want to run away from an enemy.*

### 5.43.1 Detailed Description

Forward declarations.

Contains the heuristics used by the pathfinding algorithms.

## 5.44 util::path\_type Namespace Reference

Contains different path types, which are used to check if a path should be returned once found or when an augmenting edge is found to the path.

### Classes

- struct [BEST\\_PATH](#)  
*Finds the best path by refusing any paths found.*
- struct [FIRST\\_PATH](#)  
*Finds the first path by accepting the first path found.*
- struct [RANDOM\\_PATH](#)  
*Finds a random path by returning true only when a random number in the range (0, UPPER) is equal to 0.*

### 5.44.1 Detailed Description

Contains different path types, which are used to check if a path should be returned once found or when an augmenting edge is found to the path.

## 5.45 util::pathfinding Namespace Reference

Contains the pathfinding algorithms used by the [util::pathfind](#) function.

### Classes

- struct [A\\_STAR](#)  
*Simple A\* pathfinding implementations with path type specified as a template parameter.*

### 5.45.1 Detailed Description

Contains the pathfinding algorithms used by the [util::pathfind](#) function.

## Chapter 6

# Class Documentation

### 6.1 util::pathfinding::A\_STAR< PATH\_TYPE > Struct Template Reference

Simple A\* pathfinding implementations with path type specified as a template parameter.

```
#include <PathfindingAlgorithms.hpp>
```

#### Static Public Member Functions

- static std::deque< tdt::uint > **get\_path** ([EntitySystem](#) &ents, tdt::uint id, tdt::uint start, tdt::uint end, [util::heuristic::HEURISTIC](#) &heuristic, bool allow\_destruction=true)

#### 6.1.1 Detailed Description

```
template<typename PATH_TYPE = util::DEFAULT_PATH_TYPE>
struct util::pathfinding::A_STAR< PATH_TYPE >
```

Simple A\* pathfinding implementations with path type specified as a template parameter.

##### Parameters

<i>Entity</i>	system containing the pathfinding entity and the grid.
<i>ID</i>	of the pathfinding entity.
<i>ID</i>	of the starting node.
<i>ID</i>	of the ending node.
<i>Heuristic</i>	to be used.
<i>If</i>	true, the entity will be allowed to destroy blocks along it's way.

Definition at line 42 of file PathfindingAlgorithms.hpp.

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

- tools/PathfindingAlgorithms.hpp

## 6.2 AIComponent Struct Reference

Holds info about the base Lua table to be called for initializing, updating and finishing an entity as well as couple categorizing enums.

```
#include <Components.hpp>
```

### Public Member Functions

- **AIComponent** (std::string &&s="ERROR", ENTITY\_STATE::VAL st=ENTITY\_STATE::NORMAL)
- **AIComponent** (const [AIComponent](#) &)=default
- **AIComponent** ([AIComponent](#) &&)=default
- [AIComponent](#) & **operator=** (const [AIComponent](#) &)=default
- [AIComponent](#) & **operator=** ([AIComponent](#) &&)=default

### Public Attributes

- std::string **blueprint**
- ENTITY\_STATE::VAL **state**

### Static Public Attributes

- static constexpr int **type** = 2

#### 6.2.1 Detailed Description

Holds info about the base Lua table to be called for initializing, updating and finishing an entity as well as couple categorizing enums.

Definition at line 80 of file Components.hpp.

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

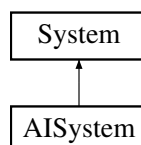
- Components.hpp

## 6.3 AISystem Class Reference

[System](#) handling the AI of entities by calling their update method every frame.

```
#include <AISystem.hpp>
```

Inheritance diagram for AISystem:



## Public Member Functions

- [AISystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~AISystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Updates all valid entities by calling their update function stored in the AIComponent::blueprint table.*
- void [set\\_update\\_period](#) (tdt::real)  
*Sets the amount of seconds it takes before the next AI update will be performed.*
- tdt::real [get\\_update\\_period](#) () const  
*Returns the amount of seconds it takes before the next AI update will be performed.*
- void [force\\_update](#) ()  
*Sets the update timer equal to the period and thus forcing all entities' AI to be updated on next [AISystem::update](#) call.*

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- tdt::real [update\\_timer\\_](#)  
*Used to track the time and check if the entities should be updated.*
- tdt::real [update\\_period\\_](#)

### 6.3.1 Detailed Description

[System](#) handling the AI of entities by calling their update method every frame.

Definition at line 10 of file AISystem.hpp.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 AISystem::AISystem ( [EntitySystem](#) & *ent* )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 6 of file AISystem.cpp.

#### 6.3.2.2 AISystem::~AISystem ( ) [default]

Destructor.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 void AISystem::force\_update ( )

Sets the update timer equal to the period and thus forcing all entities' AI to be updated on next [AISystem::update](#) call.

Definition at line 42 of file AISystem.cpp.

#### 6.3.3.2 tdt::real AISystem::get\_update\_period ( ) const

Returns the amount of seconds it takes before the next AI update will be performed.

Definition at line 37 of file AISystem.cpp.

#### 6.3.3.3 void AISystem::set\_update\_period ( tdt::real val )

Sets the amount of seconds it takes before the next AI update will be performed.

##### Parameters

<i>Update</i>	period time (in seconds).
---------------	---------------------------

Definition at line 32 of file AISystem.cpp.

#### 6.3.3.4 void AISystem::update ( tdt::real delta ) [override],[virtual]

Updates all valid entities by calling their update function stored in the AIComponent::blueprint table.

##### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 10 of file AISystem.cpp.

### 6.3.4 Member Data Documentation

#### 6.3.4.1 EntitySystem& AISystem::entities\_ [private]

Reference to the game's entity system.

Definition at line 54 of file AISystem.hpp.

## 6.3.4.2 tdt::real AISystem::update\_timer\_ [private]

Used to track the time and check if the entities should be updated.

Definition at line 59 of file AISystem.hpp.

The documentation for this class was generated from the following files:

- systems/AISystem.hpp
- systems/AISystem.cpp

## 6.4 AlignComponent Struct Reference

Holds information about an objects align states, i.e.

```
#include <Components.hpp>
```

### Classes

- struct [AlignState](#)

### Public Member Functions

- **AlignComponent** (const [AlignComponent](#) &)=default
- **AlignComponent** ([AlignComponent](#) &&)=default
- [AlignComponent](#) & **operator=** (const [AlignComponent](#) &)=default
- [AlignComponent](#) & **operator=** ([AlignComponent](#) &&)=default

### Public Attributes

- std::array< [AlignState](#), state\_count > **states**

### Static Public Attributes

- static constexpr int **type** = 24
- static constexpr int **state\_count** = 5

### 6.4.1 Detailed Description

Holds information about an objects align states, i.e.

scale, model, etc for the different alignments of blocks (e.g. walls).

Definition at line 588 of file Components.hpp.

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

- Components.hpp

## 6.5 AlignComponent::AlignState Struct Reference

### Public Attributes

- Ogre::Vector3 **scale**
- Ogre::Vector3 **position\_offset**
- std::string **mesh**
- std::string **material**

### 6.5.1 Detailed Description

Definition at line 593 of file Components.hpp.

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

- Components.hpp

## 6.6 util::path\_type::BEST\_PATH Struct Reference

Finds the best path by refusing any paths found.

```
#include <PathfindingAlgorithms.hpp>
```

### Static Public Member Functions

- static bool **return\_path** ()

### 6.6.1 Detailed Description

Finds the best path by refusing any paths found.

Definition at line 143 of file PathfindingAlgorithms.hpp.

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

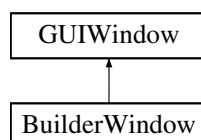
- tools/PathfindingAlgorithms.hpp

## 6.7 BuilderWindow Class Reference

Class representing the building selection window, allows the player to place registered (unlocked) buildings.

```
#include <BuilderWindow.hpp>
```

Inheritance diagram for BuilderWindow:





## Public Member Functions

- [BuilderWindow](#) ()  
*Constructor.*
- [~BuilderWindow](#) ()=default  
*Destructor.*
- void [register\\_building](#) (const std::string &)  
*Appends a table name to the vector of all building tables.*
- void [set\\_placer](#) ([EntityPlacer](#) \*)  
*Sets the placer that is used to build.*
- const std::vector< std::string > & [get\\_buildings](#) () const  
*Returns a vector containing all the names of the registered buildings.*
- void [clear\\_buildings](#) ()  
*Removes all unlocked buildings.*
- void [build](#) (int)  
*Places the building on a given position.*
- void [dec\\_selection](#) ()  
*Decrements selection\_number\_ by one and updates the window.*
- void [inc\\_selection](#) ()  
*Increments selection\_number\_ by one and updates the window.*

## Protected Member Functions

- void [init\\_](#) () override  
*Initializes the window and subscribes events.*

## Private Member Functions

- const std::string & [get\\_building\\_](#) (std::size\_t)  
*Range checked buildings\_ index access, returns the name of the building at a given index or "UNKNOWN" if the index is out of bounds.*
- void [update\\_selection\\_](#) ()  
*Updates building names on the buttons.*

## Private Attributes

- std::vector< std::string > [buildings\\_](#)  
*Names of all registered buildings.*
- std::size\_t [selection\\_number\\_](#)  
*Number of the current rightmost selection.*
- [EntityPlacer](#) \* [placer\\_](#)  
*Placer used to build.*

## Additional Inherited Members

### 6.7.1 Detailed Description

Class representing the building selection window, allows the player to place registered (unlocked) buildings.

Definition at line 12 of file BuilderWindow.hpp.

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 BuilderWindow::BuilderWindow ( )

Constructor.

Definition at line 5 of file BuilderWindow.cpp.

### 6.7.2.2 BuilderWindow::~BuilderWindow ( ) [default]

Destructor.

## 6.7.3 Member Function Documentation

### 6.7.3.1 void BuilderWindow::build ( int *build\_num* )

Places the building on a given position.

Parameters

<i>The</i>	button position. (1-4)
------------	------------------------

Definition at line 37 of file BuilderWindow.cpp.

### 6.7.3.2 void BuilderWindow::clear\_buildings ( )

Removes all unlocked buildings.

Definition at line 30 of file BuilderWindow.cpp.

### 6.7.3.3 void BuilderWindow::dec\_selection ( )

Decrements selection\_number\_ by one and updates the window.

Definition at line 101 of file BuilderWindow.cpp.

### 6.7.3.4 const std::string & BuilderWindow::get\_building\_ ( std::size\_t *index* ) [private]

Range checked buildings\_ index access, returns the name of the building at a given index or "UNKNOWN" if the index is out of bounds.

Parameters

<i>Index</i>	of the building in the buildings_ vector.
--------------	---

Definition at line 119 of file BuilderWindow.cpp.

**6.7.3.5** `const std::vector< std::string > & BuilderWindow::get_buildings ( ) const`

Returns a vector containing all the names of the registered buildings.

(Used for serialization.)

Definition at line 25 of file BuilderWindow.cpp.

**6.7.3.6** `void BuilderWindow::inc_selection ( )`

Increments selection\_number\_ by one and updates the window.

Definition at line 110 of file BuilderWindow.cpp.

**6.7.3.7** `void BuilderWindow::init_ ( ) [override], [protected], [virtual]`

Initializes the window and subscribes events.

Implements [GUIWindow](#).

Definition at line 48 of file BuilderWindow.cpp.

**6.7.3.8** `void BuilderWindow::register_building ( const std::string & tname )`

Appends a table name to the vector of all building tables.

Parameters

<i>Name</i>	of the table to register.
-------------	---------------------------

Definition at line 9 of file BuilderWindow.cpp.

**6.7.3.9** `void BuilderWindow::set_placer ( EntityPlacer * p )`

Sets the placer that is used to build.

Parameters

<i>The</i>	new entity placer.
------------	--------------------

Definition at line 20 of file BuilderWindow.cpp.

#### 6.7.3.10 void BuilderWindow::update\_selection\_( ) [private]

Updates building names on the buttons.

Definition at line 129 of file BuilderWindow.cpp.

### 6.7.4 Member Data Documentation

#### 6.7.4.1 std::vector<std::string> BuilderWindow::buildings\_ [private]

Names of all registered buildings.

Definition at line 89 of file BuilderWindow.hpp.

#### 6.7.4.2 EntityPlacer\* BuilderWindow::placer\_ [private]

Placer used to build.

Definition at line 100 of file BuilderWindow.hpp.

#### 6.7.4.3 std::size\_t BuilderWindow::selection\_number\_ [private]

Number of the current rightmost selection.

The window shows buildings with indices <selection\_number\_ - 3, selection\_number\_>.

Definition at line 95 of file BuilderWindow.hpp.

The documentation for this class was generated from the following files:

- gui/BuilderWindow.hpp
- gui/BuilderWindow.cpp

## 6.8 Camera Class Reference

Class wrapping the Ogre camera object, allowing RTS-like movement and switching to free mode.

```
#include <Camera.hpp>
```

## Public Member Functions

- [Camera](#) ()=default  
*Constructor.*
- [~Camera](#) ()=default  
*Destructor.*
- void [init](#) (Ogre::Camera \*)  
*Initializes the wrapper with a camera object.*
- void [set\\_position](#) (const Ogre::Vector2 &)  
*Sets the 2D position of the camera (X and Z axes).*
- const Ogre::Vector3 & [get\\_position](#) () const  
*Returns the 3D position of the camera (including height).*
- void [set\\_direction](#) (const Ogre::Vector3 &)  
*Changes the direction the camera is facing.*
- const Ogre::Vector3 & [get\\_direction](#) () const  
*Returns the direction the camera is facing.*
- void [look\\_at](#) (const Ogre::Vector2 &)  
*Makes the camera to look at a point on the ground.*
- void [reset](#) ()  
*Resets the camera's position and orientation.*
- void [set\\_start](#) (const Ogre::Vector2 &, const Ogre::Vector2 &, tdt::real)  
*Sets the starting stats of the camera.*
- void [set\\_free\\_mode](#) (bool)  
*Changes the movement mode of the camera.*
- bool [get\\_free\\_mode](#) () const  
*Returns true if the camera is in free mode, false otherwise.*
- void [update](#) (tdt::real)  
*Updates the movement of the camera.*
- void [key\\_pressed](#) (CEGUI::Key::Scan)  
*Moves the camera if a movement key was pressed.*
- void [key\\_released](#) (CEGUI::Key::Scan)  
*Moves the camera if a movement key was released.*
- void [move](#) (DIRECTION::VAL, tdt::real)  
*Moves the camera in a given direction, used for mouse movement.*
- void [set\\_height](#) (tdt::real)  
*Changes the height of the camera.*
- tdt::real [get\\_height](#) () const  
*Returns the height of the camera.*
- void [pitch](#) (const Ogre::Degree &)  
*Rotates the camera around the side-to-side axis.*
- void [yaw](#) (const Ogre::Degree &)  
*Rotates the camera around the vertical axis.*

## Private Attributes

- Ogre::Camera \* [camera\\_](#)  
*Ogre camera that is wrapped.*
- std::tuple< Ogre::Vector3, Ogre::Vector3 > [start\\_](#)  
*Starting stats of the camera, used when resetting.*
- bool [free\\_mode\\_](#)

- Determines the mode of the camera, if true, it can fly around the level, if false, it can move in an RTS-like fashion.*
  - `Ogre::Vector3` [movement\\_direction\\_](#)  
*Direction vector of the free mode movement.*
  - `tdt::real` [speed\\_](#)  
*Speed modifier of the camera.*
  - `tdt::real` [height\\_](#)  
*Y axis the camera is locked at when free mode is disabled.*

## Friends

- class **GameSerializer**
- class **Game**

### 6.8.1 Detailed Description

Class wrapping the Ogre camera object, allowing RTS-like movement and switching to free mode.

Definition at line 13 of file Camera.hpp.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 `Camera::Camera ( )` [default]

Constructor.

#### 6.8.2.2 `Camera::~~Camera ( )` [default]

Destructor.

### 6.8.3 Member Function Documentation

#### 6.8.3.1 `const Ogre::Vector3 & Camera::get_direction ( ) const`

Returns the direction the camera is facing.

Definition at line 27 of file Camera.cpp.

#### 6.8.3.2 `bool Camera::get_free_mode ( ) const`

Returns true if the camera is in free mode, false otherwise.

Definition at line 60 of file Camera.cpp.

#### 6.8.3.3 tdt::real Camera::get\_height ( ) const

Returns the height of the camera.

Definition at line 163 of file Camera.cpp.

#### 6.8.3.4 const Ogre::Vector3 & Camera::get\_position ( ) const

Returns the 3D position of the camera (including height).

Definition at line 16 of file Camera.cpp.

#### 6.8.3.5 void Camera::init ( Ogre::Camera \* *cam* )

Initializes the wrapper with a camera object.

##### Parameters

<i>Camera</i>	to be wrapped.
---------------	----------------

Definition at line 4 of file Camera.cpp.

#### 6.8.3.6 void Camera::key\_pressed ( CEGUI::Key::Scan *key* )

Moves the camera if a movement key was pressed.

##### Parameters

<i>Pressed</i>	<i>key</i> .
----------------	--------------

Definition at line 82 of file Camera.cpp.

#### 6.8.3.7 void Camera::key\_released ( CEGUI::Key::Scan *key* )

Moves the camera if a movement key was released.

##### Parameters

<i>Released</i>	<i>key</i> .
-----------------	--------------

Definition at line 107 of file Camera.cpp.

#### 6.8.3.8 void Camera::look\_at ( const Ogre::Vector2 & *val* )

Makes the camera to look at a point on the ground.

**Parameters**

<i>2D</i>	location of the point (X and Z axes).
-----------	---------------------------------------

Definition at line 32 of file Camera.cpp.

**6.8.3.9 void Camera::move ( DIRECTION::VAL *dir*, tdt::real *delta* )**

Moves the camera in a given direction, used for mouse movement.

**Parameters**

<i>Direction</i>	of the movement.
<i>Time</i>	since the last frame.

Definition at line 132 of file Camera.cpp.

**6.8.3.10 void Camera::pitch ( const Ogre::Degree & *val* )**

Rotates the camera around the side-to-side axis.

**Parameters**

<i>Amount</i>	of degrees to rotate by.
---------------	--------------------------

Definition at line 168 of file Camera.cpp.

**6.8.3.11 void Camera::reset ( )**

Resets the camera's position and orientation.

Definition at line 38 of file Camera.cpp.

**6.8.3.12 void Camera::set\_direction ( const Ogre::Vector3 & *val* )**

Changes the direction the camera is facing.

**Parameters**

<i>The</i>	new direction vector.
------------	-----------------------

Definition at line 21 of file Camera.cpp.



**6.8.3.13 void Camera::set\_free\_mode ( bool *val* )**

Changes the movement mode of the camera.

**Parameters**

<i>If</i>	true, the camera will be in free mode, otherwise it will return to RTS mode.
-----------	--

Definition at line 53 of file Camera.cpp.

**6.8.3.14 void Camera::set\_height ( tdt::real *val* )**

Changes the height of the camera.

**Parameters**

<i>The</i>	new height.
------------	-------------

Definition at line 154 of file Camera.cpp.

**6.8.3.15 void Camera::set\_position ( const Ogre::Vector2 & *val* )**

Sets the 2D position of the camera (X and Z axes).

**Parameters**

<i>The</i>	new position.
------------	---------------

Definition at line 10 of file Camera.cpp.

**6.8.3.16 void Camera::set\_start ( const Ogre::Vector2 & *position*, const Ogre::Vector2 & *center*, tdt::real *height* )**

Sets the starting stats of the camera.

(Used upon reset.)

**Parameters**

<i>Starting</i>	position.
<i>Starting</i>	point that camera is looking at.
<i>Starting</i>	height of the camera.

Definition at line 46 of file Camera.cpp.

#### 6.8.3.17 void Camera::update ( tdt::real *delta* )

Updates the movement of the camera.

##### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Definition at line 65 of file Camera.cpp.

#### 6.8.3.18 void Camera::yaw ( const Ogre::Degree & *val* )

Rotates the camera around the vertical axis.

##### Parameters

<i>Amount</i>	of degrees to rotate by.
---------------	--------------------------

Definition at line 173 of file Camera.cpp.

### 6.8.4 Member Data Documentation

#### 6.8.4.1 Ogre::Camera\* Camera::camera\_ [private]

Ogre camera that is wrapped.

Definition at line 142 of file Camera.hpp.

#### 6.8.4.2 bool Camera::free\_mode\_ [private]

Determines the mode of the camera, if true, it can fly around the level, if false, it can move in an RTS-like fashion.

Definition at line 154 of file Camera.hpp.

#### 6.8.4.3 tdt::real Camera::height\_ [private]

Y axis the camera is locked at when free mode is disabled.

Definition at line 170 of file Camera.hpp.

#### 6.8.4.4 Ogre::Vector3 Camera::movement\_direction\_ [private]

Direction vector of the free mode movement.

Definition at line 159 of file Camera.hpp.

## 6.8.4.5 tdt::real Camera::speed\_ [private]

Speed modifier of the camera.

Definition at line 164 of file Camera.hpp.

## 6.8.4.6 std::tuple&lt;Ogre::Vector3, Ogre::Vector3&gt; Camera::start\_ [private]

Starting stats of the camera, used when resetting.

Definition at line 147 of file Camera.hpp.

The documentation for this class was generated from the following files:

- tools/Camera.hpp
- tools/Camera.cpp

## 6.9 CombatComponent Struct Reference

Holds info about an entity's attack types and damage.

```
#include <Components.hpp>
```

### Public Member Functions

- **CombatComponent** (tdt::uint target=Component::NO\_ENTITY, tdt::uint mi=0, tdt::uint ma=0, tdt::real cd=0, tdt::real r=0.f, int type=0, bool p=false, std::string &&proj="ERROR")
- **CombatComponent** (const [CombatComponent](#) &)=default
- **CombatComponent** ([CombatComponent](#) &&)=default
- [CombatComponent](#) & **operator=** (const [CombatComponent](#) &)=default
- [CombatComponent](#) & **operator=** ([CombatComponent](#) &&)=default

### Public Attributes

- tdt::uint **curr\_target**
- tdt::uint **min\_dmg**
- tdt::uint **max\_dmg**
- tdt::real **cd\_time**
- tdt::real **cooldown**
- tdt::real **range**
- ATTACK\_TYPE **atk\_type**
- bool **pursue**
- std::string **projectile\_blueprint**

### Static Public Attributes

- static constexpr int **type** = 5

### 6.9.1 Detailed Description

Holds info about an entity's attack types and damage.

Definition at line 152 of file Components.hpp.

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

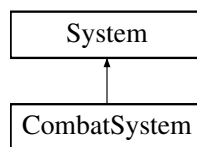
- Components.hpp

## 6.10 CombatSystem Class Reference

Manages auto attack melee and ranged combat, special melee and ranged attacks will be both handled by the spellcasting system.

```
#include <CombatSystem.hpp>
```

Inheritance diagram for CombatSystem:



### Public Member Functions

- [CombatSystem](#) ([EntitySystem](#) &, [Ogre::SceneManager](#) &, [GridSystem](#) &)  
*Constructor.*
- [~CombatSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Updates all auto attack combat in the game currently in progress.*
- bool [in\\_sight](#) (tdt::uint, tdt::uint) const  
*Returns true if two given entities can see each other, false otherwise.*
- bool [in\\_sight\\_wrt\\_BB](#) (tdt::uint, tdt::uint) const  
*Returns true if two given entities can see each other, false otherwise.*
- tdt::uint [get\\_closest\\_entity](#) (tdt::uint, bool=true, bool=false) const  
*Returns the ID of the closest entity (from a given entity's position), Component::NO\_ENTITY otherwise.*
- tdt::uint [get\\_closest\\_structure](#) (tdt::uint, bool=true, bool=false) const  
*Returns the ID of the closest structure (from a given entity's position), Component::NO\_ENTITY otherwise.*
- tdt::uint [get\\_closest\\_entity\\_thats\\_not](#) (tdt::uint, tdt::uint, bool=true, bool=false) const  
*Returns the ID of the closest entity (from a given entity's position) ignoring a given entity, Component::NO\_ENTITY otherwise.*
- tdt::uint [get\\_closest\\_gold\\_deposit](#) (tdt::uint, bool=false) const  
*Returns the ID of the closest gold deposit (entity with both structure and gold components).*
- tdt::uint [get\\_closest\\_gold\\_vault](#) (tdt::uint, bool=false, bool=false) const  
*Returns the ID of the closest gold vault that can store player's gold.*
- template<typename CONT , typename COND >  
tdt::uint [get\\_closest\\_entity](#) (tdt::uint id, COND &condition, bool only\_sight=true) const

- Returns the ID of the closest entity that has a given component, meets a given condition and is accessible.*
- `template<typename CONT , typename COND , typename EFFECT >`  
`void apply_effect_to_entities_in_range (tdt::uint id, COND &condition, EFFECT &effect, tdt::real range)`  
*Applies a given effect (functor given by the template argument EFFECT) to all entities conforming the condition (functor given by the template argument COND) in a given range from a given entity.*
- `void apply_heal_to_entities_in_range (tdt::uint, tdt::real)`  
*Heals all friendly entities within a given range from a given entity.*
- `void apply_damage_to_entities_in_range (tdt::uint, tdt::real, tdt::uint, tdt::uint)`  
*Damages all friendly entities within a given range from a given entity.*
- `void apply_slow_to_entities_in_range (tdt::uint, tdt::real, tdt::real)`  
*Slows all enemy entities within a given range from a given entity for a given time period.*
- `void apply_freeze_to_entities_in_range (tdt::uint, tdt::real, tdt::real)`  
*Freezes all enemy entities within a given range from a given entity for a given time period.*
- `void apply_slow_to (tdt::uint, tdt::real)`  
*Slows a given entity for a given time period.*
- `void apply_freeze_to (tdt::uint, tdt::real)`  
*Freezes a given entity for a given time period.*
- `void run_away_from (tdt::uint, tdt::uint, tdt::uint)`  
*Tries to find a path used by an entity to run away from another entity.*
- `void set_max_run_away_attempts (tdt::uint)`  
*Sets the maximum amount of pathfinding attempts for running away.*
- `tdt::uint get_max_run_away_attempts ()`  
*Returns the maximum amount of pathfinding attempts for running away.*
- `bool enemy_in_range (tdt::uint)`  
*Returns true if an enemy is in range from a given entity, false otherwise.*
- `template<>`  
`const std::map< tdt::uint, ALL_COMPONENTS > &get_container () const`  
*Specific case of the get\_container method, which returns the map containing <ID, component bitset> map containing all entities.*

## Private Member Functions

- `template<typename COMP >`  
`const std::map< tdt::uint, COMP > &get_container () const`  
*Returns a map containing pairs of IDs and components of a given type, use the type ALL\_COMPONENTS to get the <ID, component bitset> container.*
- `void create_homing_projectile (tdt::uint, CombatComponent &)`  
*Creates a new homing projectile at the position of a given entity homing at the entity's current target.*
- `void run_away_from_ (tdt::uint, tdt::uint, tdt::uint)`  
*Tries to find a path used by an entity to run away from another entity.*

## Private Attributes

- `EntitySystem` & `entities_`  
*Reference to the game's entity system (component retrieval).*
- `Ogre::RaySceneQuery` & `ray_query_`  
*Reference to the ray cast used to check if two entities can see each other.*
- `GridSystem` & `grid_`  
*Used to check if an entity is accessible.*
- `RayCaster` `ray_caster_`

*Used for polygon precise line of sight checking.*

- `tdt::uint max_run_away_attempts_ {10}`

*Maximum amount of pathfindings performed when running away from an enemy.*

- `std::queue< std::tuple< tdt::uint, tdt::uint, tdt::uint > > run_away_queue_`

*Used to distribute run away pathfindings over frames (otherwise something like a meteor falling on a big group of entities would freeze the game until all run away pathfindings are done.)*

### 6.10.1 Detailed Description

Manages auto attack melee and ranged combat, special melee and ranged attacks will be both handled by the spellcasting system.

Definition at line 28 of file `CombatSystem.hpp`.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 `CombatSystem::CombatSystem ( EntitySystem & ents, Ogre::SceneManager & scene, GridSystem & grid )`

Constructor.

Parameters

<i>Reference</i>	to the game's entity system (component retrieval).
<i>Reference</i>	to the main scene manager (ray casting).
<i>Reference</i>	to the game's grid system (accessibility).

Definition at line 11 of file `CombatSystem.cpp`.

#### 6.10.2.2 `CombatSystem::~~CombatSystem ( ) [default]`

Destructor.

### 6.10.3 Member Function Documentation

#### 6.10.3.1 `void CombatSystem::apply_damage_to_entities_in_range ( tdt::uint id, tdt::real range, tdt::uint min, tdt::uint max )`

Damages all friendly entities within a given range from a given entity.

Parameters

<i>ID</i>	of the entity.
<i>The</i>	range.
<i>Minimal</i>	damage value.
<i>Maximal</i>	damage value.

Definition at line 268 of file CombatSystem.cpp.

**6.10.3.2** `template<typename CONT , typename COND , typename EFFECT > void CombatSystem::apply↵  
_effect_to_entities_in_range ( tdt::uint id, COND & condition, EFFECT & effect, tdt::real range )  
[inline]`

Applies a given effect (functor given by the template argument EFFECT) to all entities conforming the condition (functor given by the template argument COND) in a given range from a given entity.

#### Parameters

<i>ID</i>	of the entity.
<i>Instance</i>	of the condition functor.
<i>Instance</i>	of the effect functor.
<i>The</i>	radius.

#### Note

The explicit template specialization determines over which component container this method will iterate, use a component name for a specific components only or ALL\_COMPONENTS for the component bitset map (which allows to iterate over all entities regardless of their components).

Definition at line 170 of file CombatSystem.hpp.

**6.10.3.3** `void CombatSystem::apply_freeze_to ( tdt::uint id, tdt::real time )`

Freezes a given entity for a given time period.

#### Parameters

<i>ID</i>	of the entity.
<i>The</i>	time period for which the freeze is active.

Definition at line 295 of file CombatSystem.cpp.

**6.10.3.4** `void CombatSystem::apply_freeze_to_entities_in_range ( tdt::uint id, tdt::real range, tdt::real time )`

Freezes all enemy entities within a given range from a given entity for a given time period.

#### Parameters

<i>ID</i>	of the entity.
<i>The</i>	range.
<i>The</i>	time period for which the freeze is active.

Definition at line 282 of file CombatSystem.cpp.

#### 6.10.3.5 void CombatSystem::apply\_heal\_to\_entities\_in\_range ( tdt::uint *id*, tdt::real *range* )

Heals all friendly entities within a given range from a given entity.

##### Parameters

<i>ID</i>	of the entity.
<i>The</i>	range.

Definition at line 261 of file CombatSystem.cpp.

#### 6.10.3.6 void CombatSystem::apply\_slow\_to ( tdt::uint *id*, tdt::real *time* )

Slows a given entity for a given time period.

##### Parameters

<i>ID</i>	of the entity.
<i>The</i>	time period for which the freeze is active.

Definition at line 289 of file CombatSystem.cpp.

#### 6.10.3.7 void CombatSystem::apply\_slow\_to\_entities\_in\_range ( tdt::uint *id*, tdt::real *range*, tdt::real *time* )

Slows all enemy entities within a given range from a given entity for a given time period.

##### Parameters

<i>ID</i>	of the entity.
<i>The</i>	range.
<i>The</i>	time period for which the slow is active.

Definition at line 275 of file CombatSystem.cpp.

#### 6.10.3.8 void CombatSystem::create\_homing\_projectile ( tdt::uint *caster*, CombatComponent & *combat* ) [private]

Creates a new homing projectile at the position of a given entity homing at the entity's current target.

##### Parameters

<i>ID</i>	of the caster entity.
<i>Reference</i>	to the caster entity's combat component.

Definition at line 387 of file CombatSystem.cpp.



6.10.3.9 `bool CombatSystem::enemy_in_range ( tdt::uint id )`

Returns true if an enemy is in range from a given entity, false otherwise.

## Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 356 of file CombatSystem.cpp.

6.10.3.10 `std::size_t CombatSystem::get_closest_entity ( tdt::uint id, bool only_sight = true, bool friendly = false ) const`

Returns the ID of the closest entity (from a given entity's position), Component::NO\_ENTITY otherwise.

## Parameters

<i>ID</i>	of the entity from whose position the search is performed.
<i>If</i>	true, will return only entities in sight.
<i>If</i>	true, will return only friendly entities (enemies otherwise).

Definition at line 208 of file CombatSystem.cpp.

6.10.3.11 `template<typename CONT, typename COND> tdt::uint CombatSystem::get_closest_entity ( tdt::uint id, COND & condition, bool only_sight = true ) const [inline]`

Returns the ID of the closest entity that has a given component, meets a given condition and is accessible.

## Parameters

<i>ID</i>	of the entity that is searching.
<i>Functor</i>	representing the condition.
<i>If</i>	true, only entities in sight get checked.

## Note

The explicit template specialization determines over which component container this method will iterate, use a component name for a specific components only or ALL\_COMPONENTS for the component bitset map (which allows to iterate over all entities regardless of their components).

Definition at line 128 of file CombatSystem.hpp.

6.10.3.12 `std::size_t CombatSystem::get_closest_entity_thats_not ( tdt::uint id, tdt::uint ignored, bool only_sight = true, bool friendly = false ) const`

Returns the ID of the closest entity (from a given entity's position) ignoring a given entity, Component::NO\_ENTITY otherwise.

## Parameters

<i>ID</i>	of the entity from whose position the search is performed.
<i>ID</i>	of the entity that's ignored in the search.
<i>If</i>	true, will return only entities in sight.
<i>If</i>	true, will return only friendly entities (enemies otherwise).

Definition at line 224 of file CombatSystem.cpp.

**6.10.3.13** `std::size_t CombatSystem::get_closest_gold_deposit ( tdt::uint id, bool only_sight = false ) const`

Returns the ID of the closest gold deposit (entity with both structure and gold components).

## Parameters

<i>ID</i>	of the entity that looks for the gold deposit.
<i>If</i>	true, only deposits in sight will be checked.

Definition at line 238 of file CombatSystem.cpp.

**6.10.3.14** `std::size_t CombatSystem::get_closest_gold_vault ( tdt::uint id, bool only_sight = false, bool only_free = false ) const`

Returns the ID of the closest gold vault that can store player's gold.

## Parameters

<i>ID</i>	of the entity that looks for the gold vault.
<i>If</i>	true, only vaults in sight will be checked.
<i>If</i>	true, only vaults that have free space for more gold will be checked.

Definition at line 247 of file CombatSystem.cpp.

**6.10.3.15** `std::size_t CombatSystem::get_closest_structure ( tdt::uint id, bool only_sight = true, bool friendly = false ) const`

Returns the ID of the closest structure (from a given entity's position), Component::NO\_ENTITY otherwise.

## Parameters

<i>ID</i>	of the entity from whose position the search is performed.
<i>If</i>	true, will return only structures in sight.
<i>If</i>	true, will return friendly structures (enemies otherwise).

Definition at line 216 of file CombatSystem.cpp.

**6.10.3.16** `template<typename COMP> const std::map<tdt::uint, COMP>& CombatSystem::get_container ( ) const`  
`[inline], [private]`

Returns a map containing pairs of IDs and components of a given type, use the type ALL\_COMPONENTS to get the <ID, component bitset> container.

Definition at line 269 of file CombatSystem.hpp.

**6.10.3.17** `template<> const std::map<tdt::uint, ALL_COMPONENTS>& CombatSystem::get_container ( ) const`  
`[inline]`

Specific case of the get\_container method, which returns the map containing <ID, component bitset> map containing all entities.

Definition at line 331 of file CombatSystem.hpp.

**6.10.3.18** `std::size_t CombatSystem::get_max_run_away_attempts ( )`

Returns the maximum amount of pathfinding attempts for running away.

Definition at line 351 of file CombatSystem.cpp.

**6.10.3.19** `bool CombatSystem::in_sight ( tdt::uint ent_id, tdt::uint target ) const`

Returns true if two given entities can see each other, false otherwise.

Tests polygons.

#### Parameters

<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity. NOTE: Tests only if entities that have query flags of WALL or BUILDING are in the way, allows to see through other friendly/enemy/neutral entities.

Definition at line 157 of file CombatSystem.cpp.

**6.10.3.20** `bool CombatSystem::in_sight_wrt_BB ( tdt::uint ent_id, tdt::uint target ) const`

Returns true if two given entities can see each other, false otherwise.

Tests only bounding boxes.

#### Parameters

<i>ID</i>	of the first entity.
<i>ID</i>	of the second entity. NOTE: Tests only if entities that have query flags of WALL or BUILDING are in the way, allows to see through other friendly/enemy/neutral entities.

Definition at line 181 of file CombatSystem.cpp.

6.10.3.21 `void CombatSystem::run_away_from ( tdt::uint id, tdt::uint from_id, tdt::uint min_node_count )`

Tries to find a path used by an entity to run away from another entity.

#### Parameters

<i>ID</i>	of the entity running away.
<i>ID</i>	of the entity that is ran away from.
<i>Minimal</i>	amount of nodes the path has to have (will be ignored if the amount of attempts surpasses the maximum amount).

#### Note

This is just a dummy that enqueues the entity for pathfinding.

Definition at line 346 of file CombatSystem.cpp.

6.10.3.22 `void CombatSystem::run_away_from_ ( tdt::uint id, tdt::uint from_id, tdt::uint min_node_count )` `[private]`

Tries to find a path used by an entity to run away from another entity.

#### Parameters

<i>ID</i>	of the entity running away.
<i>ID</i>	of the entity that is ran away from.
<i>Minimal</i>	amount of nodes the path has to have (will be ignored if the amount of attempts surpasses the maximum amount).

#### Note

This is the actual implementation.

Definition at line 301 of file CombatSystem.cpp.

6.10.3.23 `void CombatSystem::set_max_run_away_attempts ( tdt::uint val )`

Sets the maximum amount of pathfinding attempts for running away.

#### Parameters

<i>The</i>	new maximum amount.
------------	---------------------

Definition at line 341 of file CombatSystem.cpp.

**6.10.3.24** `void CombatSystem::update ( tdt::real delta )` `[override], [virtual]`

Updates all auto attack combat in the game currently in progress.

#### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Check if the target is in range and in sight and if so, attack, otherwise pursue the target if necessary.

Implements [System](#).

Definition at line 19 of file CombatSystem.cpp.

## 6.10.4 Member Data Documentation

**6.10.4.1** `EntitySystem& CombatSystem::entities_` `[private]`

Reference to the game's entity system (component retrieval).

Definition at line 296 of file CombatSystem.hpp.

**6.10.4.2** `GridSystem& CombatSystem::grid_` `[private]`

Used to check if an entity is accessible.

Definition at line 306 of file CombatSystem.hpp.

**6.10.4.3** `tdt::uint CombatSystem::max_run_away_attempts_{10}` `[private]`

Maximum amount of pathfindings performed when running away from an enemy.

Definition at line 316 of file CombatSystem.hpp.

**6.10.4.4** `RayCaster CombatSystem::ray_caster_` `[private]`

Used for polygon precise line of sight checking.

Definition at line 311 of file CombatSystem.hpp.

**6.10.4.5** `Ogre::RaySceneQuery& CombatSystem::ray_query_` `[private]`

Reference to the ray cast used to check if two entities can see each other.

Definition at line 301 of file CombatSystem.hpp.

6.10.4.6 `std::queue<std::tuple<tdt::uint, tdt::uint, tdt::uint> > CombatSystem::run_away_queue_` [private]

Used to distribute run away pathfindings over frames (otherwise something like a meteor falling on a big group of entities would freeze the game until all run away pathfindings are done.)

Definition at line 323 of file `CombatSystem.hpp`.

The documentation for this class was generated from the following files:

- `systems/CombatSystem.hpp`
- `systems/CombatSystem.cpp`

## 6.11 CommandComponent Struct Reference

Contains a list of commands an entity can respond to.

```
#include <Components.hpp>
```

### Public Member Functions

- **CommandComponent** (const [CommandComponent](#) &)=default
- **CommandComponent** ([CommandComponent](#) &&)=default
- [CommandComponent](#) & **operator=** (const [CommandComponent](#) &)=default
- [CommandComponent](#) & **operator=** ([CommandComponent](#) &&)=default

### Public Attributes

- `std::bitset<(int) COMMAND_TYPE::COUNT > possible_commands`

### Static Public Attributes

- static constexpr int **type** = 37

#### 6.11.1 Detailed Description

Contains a list of commands an entity can respond to.

Definition at line 859 of file `Components.hpp`.

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

- `Components.hpp`

## 6.12 Component Struct Reference

### Static Public Attributes

- static constexpr int **count** = 40
- static constexpr tdt::uint **NO\_ENTITY** = std::numeric\_limits<tdt::uint>::max()

### 6.12.1 Detailed Description

Definition at line 14 of file Components.hpp.

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

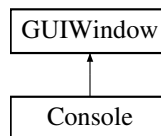
- Components.hpp

## 6.13 Console Class Reference

Class representing the ingame developers console that allows for runtime execution of Lua code.

```
#include <Console.hpp>
```

Inheritance diagram for Console:



### Public Member Functions

- [Console](#) ()  
*Constructor.*
- [~Console](#) ()=default  
*Destructor.*
- void [set\\_visible](#) (bool) override  
*Changes the visibility and text capturing of the console window.*
- bool [handle\\_text](#) (const CEGUI::EventArgs &)  
*Event handler that is called by CEGUI whenever a text is entered.*
- bool [execute](#) (const CEGUI::EventArgs &)  
*Event handler that is called by CEGUI whenever the EXECUTE button is pressed.*
- void [print\\_text](#) (const std::string &, CEGUI::Colour=CEGUI::Colour{0xFFFFFFFF})  
*Prints a given message into the console window.*
- void [scroll\\_down](#) (tdt::uint=1)  
*Scrolls the console output down one line, used so that messages from the outside can scroll all the way down to the last line of the output.*
- void [update\\_fps](#) (tdt::real, tdt::real)  
*Updates the FPS value next to the console name.*
- void [set\\_history](#) (tdt::uint)  
*Sets the number of entries that will be shown in the console's history.*
- tdt::uint [get\\_history](#) () const  
*Returns the number of entries that will be shown in the console's history.*
- void [clear](#) ()  
*Clears the console log.*

## Static Public Attributes

- static const CEGUI::Colour **RED\_TEXT** = CEGUI::Colour{1.f, 0.f, 0.f}
- static const CEGUI::Colour **GREEN\_TEXT** = CEGUI::Colour{0.f, 1.f, 0.f}
- static const CEGUI::Colour **ORANGE\_TEXT** = CEGUI::Colour{1.f, .5f, 0.1f}
- static const CEGUI::Colour **BLUE\_TEXT** = CEGUI::Colour{0.f, 0.f, 1.f}

## Protected Member Functions

- void [init\\_\(\)](#) override  
*Initializes the console and subscribes it to events.*

## Private Attributes

- CEGUI::Listbox \* [list\\_box\\_](#)  
*Pointer to the CEGUI ListBox widget that serves as console output.*
- std::string [curr\\_command\\_](#)  
*String containing the current command, allows to append more statements to it until the EXECUTE button is pressed, which allows for multi-line Lua code.*
- tdt::real [time\\_since\\_last\\_fps\\_update\\_](#)  
*Monitors the time passed since the fps label was last updated, used to make sure the fps update won't slow down the game as it involves float to string conversion.*
- tdt::uint [console\\_history\\_](#)  
*Limits the number of console entries that will be shown in it's history.*

## Additional Inherited Members

### 6.13.1 Detailed Description

Class representing the ingame developers console that allows for runtime execution of Lua code.

Definition at line 12 of file Console.hpp.

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 Console::Console ( )

Constructor.

Definition at line 10 of file Console.cpp.

#### 6.13.2.2 Console::~~Console ( ) [default]

Destructor.

### 6.13.3 Member Function Documentation

#### 6.13.3.1 void Console::clear ( )

Clears the console log.

Definition at line 130 of file Console.cpp.

#### 6.13.3.2 bool Console::execute ( const CEGUI::EventArgs & args )

Event handler that is called by CEGUI whenever the EXECUTE button is pressed.



## Parameters

<i>Reference</i>	to the CEGUI argument.
------------------	------------------------

Definition at line 44 of file Console.cpp.

#### 6.13.3.3 tdt::uint Console::get\_history ( ) const

Returns the number of entries that will be shown in the console's history.

Definition at line 125 of file Console.cpp.

#### 6.13.3.4 bool Console::handle\_text ( const CEGUI::EventArgs & args )

Event handler that is called by CEGUI whenever a text is entered.

## Parameters

<i>Reference</i>	to the CEGUI argument.
------------------	------------------------

Definition at line 35 of file Console.cpp.

#### 6.13.3.5 void Console::init\_ ( ) [override], [protected], [virtual]

Initializes the console and subscribes it to events.

Implements [GUIWindow](#).

Definition at line 15 of file Console.cpp.

#### 6.13.3.6 void Console::print\_text ( const std::string & msg, CEGUI::Colour col = CEGUI::Colour{0xFFFFFFFF} )

Prints a given message into the console window.

## Parameters

<i>Message</i>	to be printed.
<i>Colour</i>	of the message text, defaults to white.

Definition at line 82 of file Console.cpp.

#### 6.13.3.7 void Console::scroll\_down ( tdt::uint num\_of\_scrolls = 1 )

Scrolls the console output down one line, used so that messages from the outside can scroll all the way down to the last line of the output.

## Parameters

<i>Amount</i>	of lines to scroll.
---------------	---------------------

Definition at line 104 of file Console.cpp.

#### 6.13.3.8 void Console::set\_history ( tdt::uint *val* )

Sets the number of entries that will be shown in the console's history.

## Parameters

<i>The</i>	new entry count.
------------	------------------

Definition at line 120 of file Console.cpp.

#### 6.13.3.9 void Console::set\_visible ( bool *visible* ) [override], [virtual]

Changes the visibility and text capturing of the console window.

## Parameters

<i>The</i>	new visibility state.
------------	-----------------------

Reimplemented from [GUIWindow](#).

Definition at line 28 of file Console.cpp.

#### 6.13.3.10 void Console::update\_fps ( tdt::real *delta*, tdt::real *fps* )

Updates the FPS value next to the console name.

## Parameters

<i>Time</i>	since the last frame.
<i>The</i>	new framerate.

Definition at line 110 of file Console.cpp.

### 6.13.4 Member Data Documentation

#### 6.13.4.1 tdt::uint Console::console\_history\_ [private]

Limits the number of console entries that will be shown in it's history.

Definition at line 110 of file Console.hpp.

6.13.4.2 `std::string Console::curr_command_` [private]

String containing the current command, allows to append more statements to it until the EXECUTE button is pressed, which allows for multi-line Lua code.

Definition at line 98 of file Console.hpp.

6.13.4.3 `CEGUI::Listbox* Console::list_box_` [private]

Pointer to the CEGUI ListBox widget that serves as console output.

Definition at line 91 of file Console.hpp.

6.13.4.4 `ttdt::real Console::time_since_last_fps_update_` [private]

Monitors the time passed since the fps label was last updated, used to make sure the fps update won't slow down the game as it involves float to string conversion.

Definition at line 105 of file Console.hpp.

The documentation for this class was generated from the following files:

- gui/Console.hpp
- gui/Console.cpp

## 6.14 ConstructorComponent Struct Reference

Contains the blueprint that gets called when an entity that has this component is created.

```
#include <Components.hpp>
```

### Public Member Functions

- **ConstructorComponent** (std::string &&b="ERROR")
- **ConstructorComponent** (const [ConstructorComponent](#) &)=default
- **ConstructorComponent** ([ConstructorComponent](#) &&)=default
- [ConstructorComponent](#) & **operator=** (const [ConstructorComponent](#) &)=default
- [ConstructorComponent](#) & **operator=** ([ConstructorComponent](#) &&)=default

### Public Attributes

- std::string **blueprint**

### Static Public Attributes

- static constexpr int **type** = 28

### 6.14.1 Detailed Description

Contains the blueprint that gets called when an entity that has this component is created.

Definition at line 669 of file Components.hpp.

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

- Components.hpp

## 6.15 CounterComponent Struct Reference

A simple incrementing counter.

```
#include <Components.hpp>
```

### Public Member Functions

- **CounterComponent** (tdt::uint max=0)
- **CounterComponent** (const [CounterComponent](#) &)=default
- **CounterComponent** ([CounterComponent](#) &&)=default
- [CounterComponent](#) & **operator=** (const [CounterComponent](#) &)=default
- [CounterComponent](#) & **operator=** ([CounterComponent](#) &&)=default

### Public Attributes

- tdt::uint **curr\_value**
- tdt::uint **max\_value**

### Static Public Attributes

- static constexpr int **type** = 38

### 6.15.1 Detailed Description

A simple incrementing counter.

#### Note

The max value is not enforced and serves only for manual checking.

Definition at line 878 of file Components.hpp.

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

- Components.hpp

## 6.16 util::effect::DAMAGE\_EFFECT Struct Reference

Deals a random damage in a given range to the entity it's called on.

```
#include <Effects.hpp>
```

### Public Member Functions

- [DAMAGE\\_EFFECT](#) ([EntitySystem](#) &, tdt::uint, tdt::uint)  
*Constructor.*
- [~DAMAGE\\_EFFECT](#) ()=default  
*Destructor.*
- void [operator\(\)](#) (tdt::uint)  
*Applies the damage effect to a given entity.*

### Private Attributes

- tdt::uint [min\\_](#)  
*The damage range.*
- tdt::uint [max\\_](#)
- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing the entities this effect will be used on.*

#### 6.16.1 Detailed Description

Deals a random damage in a given range to the entity it's called on.

Definition at line 23 of file Effects.hpp.

#### 6.16.2 Constructor & Destructor Documentation

##### 6.16.2.1 util::effect::DAMAGE\_EFFECT::DAMAGE\_EFFECT ( [EntitySystem](#) & *ents*, tdt::uint *min*, tdt::uint *max* )

Constructor.

Parameters

<i>Entity</i>	system containing the entities this effect will be used on.
<i>Lower</i>	bound of the damage range.
<i>Upper</i>	bound of the damage range.

Definition at line 7 of file Effects.cpp.

##### 6.16.2.2 util::effect::DAMAGE\_EFFECT::~~DAMAGE\_EFFECT ( ) [default]

Destructor.

### 6.16.3 Member Function Documentation

#### 6.16.3.1 void util::effect::DAMAGE\_EFFECT::operator() ( tdt::uint *id* )

Applies the damage effect to a given entity.

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 11 of file Effects.cpp.

### 6.16.4 Member Data Documentation

#### 6.16.4.1 EntitySystem& util::effect::DAMAGE\_EFFECT::entities\_ [private]

Entity system containing the entities this effect will be used on.

Definition at line 55 of file Effects.hpp.

#### 6.16.4.2 tdt::uint util::effect::DAMAGE\_EFFECT::min\_ [private]

The damage range.

Definition at line 49 of file Effects.hpp.

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

- tools/Effects.hpp
- tools/Effects.cpp

## 6.17 DestructorComponent Struct Reference

Contains name of the table that contains the function (called "dtor") which is called when an entity is destroyed.

```
#include <Components.hpp>
```

### Public Member Functions

- **DestructorComponent** (std::string b="ERROR")
- **DestructorComponent** (const [DestructorComponent](#) &)=default
- **DestructorComponent** ([DestructorComponent](#) &&)=default
- [DestructorComponent](#) & **operator=** (const [DestructorComponent](#) &)=default
- [DestructorComponent](#) & **operator=** ([DestructorComponent](#) &&)=default

## Public Attributes

- `std::string` **blueprint**

## Static Public Attributes

- `static constexpr int` **type** = 20

### 6.17.1 Detailed Description

Contains name of the table that contains the function (called "dtor") which is called when an entity is destroyed.

Definition at line 508 of file Components.hpp.

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

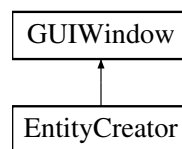
- Components.hpp

## 6.18 EntityCreator Class Reference

Class representing the debugging GUI window used to place and create entities during runtime.

```
#include <EntityCreator.hpp>
```

Inheritance diagram for EntityCreator:



## Public Member Functions

- `EntityCreator` (`EntityPlacer` &, `EntitySystem` &)  
*Constructor, loads the gui layout for this window and registers callbacks.*
- `~EntityCreator` ()  
*Destructor.*
- `bool place` (const `CEGUI::EventArgs` &)  
*Function that is called when the player/developer presses the "place now" button, sets the currently selected entity blueprint for placing.*
- `bool change_to_place` (const `CEGUI::EventArgs` &)  
*Function that is called when the player/developer presses the "place" button, changing the creator to the placing mode.*
- `bool change_to_create` (const `CEGUI::EventArgs` &)  
*Function that is called when the player/developer presses the "create" button, changing the creator to the creation mode.*
- `bool actualize_list` (const `CEGUI::EventArgs` &)  
*Function that is called when the player/developer presses the "actualize list" button and displays the set of all selected entities to the list box.*

## Protected Member Functions

- void `init_()` override  
*Initializes the [EntityCreator](#).*

## Private Attributes

- [EntityPlacer](#) & `placer_`  
*Reference to the game's entity placer, used to set the blueprint table and visibility mode when the player/developer presses the "place now" button.*
- `std::set< std::string >` & [registered\\_entities\\_](#)  
*Reference to the list of all registered entity blueprint names used for updates.*
- `CEGUI::Listbox *` [list\\_box\\_](#)  
*Auxiliary pointer to the list box sub window for easy access when updating the entity blueprint list.*

## Additional Inherited Members

### 6.18.1 Detailed Description

Class representing the debugging [GUI](#) window used to place and create entities during runtime.

Definition at line 14 of file `EntityCreator.hpp`.

### 6.18.2 Constructor & Destructor Documentation

#### 6.18.2.1 `EntityCreator::EntityCreator ( EntityPlacer & placer, EntitySystem & ents )`

Constructor, loads the gui layout for this window and registers callbacks.

##### Parameters

<i>Reference</i>	to the game's entity placer.
<i>Reference</i>	to the game's entity system.

Definition at line 6 of file `EntityCreator.cpp`.

#### 6.18.2.2 `EntityCreator::~EntityCreator ( )` `[inline]`

Destructor.

Definition at line 27 of file `EntityCreator.hpp`.

### 6.18.3 Member Function Documentation

#### 6.18.3.1 `bool EntityCreator::actualize_list ( const CEGUI::EventArgs & args )`

Function that is called when the player/developer presses the "actualize list" button and displays the set of all selected entities to the list box.



## Parameters

<i>Reference</i>	to the CEGUI event arguments.
------------------	-------------------------------

Definition at line 34 of file EntityCreator.cpp.

#### 6.18.3.2 bool EntityCreator::change\_to\_create ( const CEGUI::EventArgs & args )

Function that is called when the player/developer presses the "create" button, changing the creator to the creation mode.

## Parameters

<i>Reference</i>	to the CEGUI event arguments.
------------------	-------------------------------

Definition at line 28 of file EntityCreator.cpp.

#### 6.18.3.3 bool EntityCreator::change\_to\_place ( const CEGUI::EventArgs & args )

Function that is called when the player/developer presses the "place" button, changing the creator to the placing mode.

## Parameters

<i>Reference</i>	to the CEGUI event arguments.
------------------	-------------------------------

Definition at line 22 of file EntityCreator.cpp.

#### 6.18.3.4 void EntityCreator::init\_ ( ) [override], [protected], [virtual]

Initializes the [EntityCreator](#).

Implements [GUIWindow](#).

Definition at line 51 of file EntityCreator.cpp.

#### 6.18.3.5 bool EntityCreator::place ( const CEGUI::EventArgs & args )

Function that is called when the player/developer presses the "place now" button, sets the currently selected entity blueprint for placing.

## Parameters

<i>Reference</i>	to the CEGUI event arguments.
------------------	-------------------------------

Definition at line 10 of file EntityCreator.cpp.

### 6.18.4 Member Data Documentation

#### 6.18.4.1 CEGUI::Listbox\* EntityCreator::list\_box\_ [private]

Auxiliary pointer to the list box sub window for easy access when updating the entity blueprint list.

Definition at line 80 of file EntityCreator.hpp.

#### 6.18.4.2 EntityPlacer& EntityCreator::placer\_ [private]

Reference to the game's entity placer, used to set the blueprint table and visibility mode when the player/developer presses the "place now" button.

Definition at line 69 of file EntityCreator.hpp.

#### 6.18.4.3 std::set<std::string>& EntityCreator::registered\_entities\_ [private]

Reference to the list of all registered entity blueprint names used for updates.

Definition at line 74 of file EntityCreator.hpp.

The documentation for this class was generated from the following files:

- gui/EntityCreator.hpp
- gui/EntityCreator.cpp

## 6.19 util::EntityDestroyer Class Reference

A structure providing the private method [EntitySystem::destroy\\_entity](#) to the [DestructorHelper::destroy](#) function.

```
#include <Util.hpp>
```

### Static Private Member Functions

- static void [destroy](#) ([EntitySystem](#) &, tdt::uint)  
*Destroy a given entity.*

### Friends

- void [DestructorHelper::destroy](#) ([EntitySystem](#) &, tdt::uint, bool, tdt::uint)

### 6.19.1 Detailed Description

A structure providing the private method [EntitySystem::destroy\\_entity](#) to the [DestructorHelper::destroy](#) function.

The reason for the existence of this struct is that it provides only this one method and keeps others private.

Definition at line 194 of file Util.hpp.

### 6.19.2 Member Function Documentation

#### 6.19.2.1 void util::EntityDestroyer::destroy ( [EntitySystem](#) & *ents*, tdt::uint *id* ) [static], [private]

Destroy a given entity.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 40 of file Util.cpp.

The documentation for this class was generated from the following files:

- tools/Util.hpp
- tools/Util.cpp

## 6.20 EntityPlacer Class Reference

Class allowing the player/developer to place entities on the ground.

```
#include <EntityPlacer.hpp>
```

### Public Member Functions

- [EntityPlacer](#) ([EntitySystem](#) &, [GridSystem](#) &, [Ogre::SceneManager](#) &)  
*Constructor.*
- [~EntityPlacer](#) ()  
*Destructor.*
- void [set\\_current\\_entity\\_table](#) (const std::string &, bool=false)  
*Checks if a given entity blueprint contains a graphics component and if so, sets it as the currently placed entity by creating a dummy entity following the cursor.*
- void [update\\_position](#) (const [Ogre::Vector3](#) &)  
*Called every frame when the entity placer is active, adjusts the dummy entity's position to the cursor's.*
- [tdt::uint](#) [place](#) ()  
*Creates a new entity from the blueprint table at the mouse cursor's current position and informs the developer in the developer console.*
- void [set\\_visible](#) (bool)  
*Sets the visibility status of the placer (and it's dummy entity).*
- bool [is\\_visible](#) () const  
*Returns true if the placer is currently active (and thus the dummy entity visible), false otherwise.*
- void [toggle\\_placing\\_when\\_game\\_paused](#) ()  
*Toggles (=negates) the flag for entity placing while the game is paused.*
- bool [can\\_place\\_when\\_game\\_paused](#) () const  
*Returns true if the placer can place entities when the game is paused, false otherwise.*

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- [GridSystem](#) & [grid\\_](#)  
*Reference to the game's grid system.*
- [Ogre::Vector3](#) [curr\\_position\\_](#)  
*Current position of the dummy entity (used to avoid the need to pass the position again on the actual placement call).*
- [Ogre::SceneNode](#) \* [placing\\_node\\_](#)  
*Pointer to the scene node the dummy entity is attached to.*
- [bool](#) [visible\\_](#)  
*The placer's visibility (and active) status.*
- [std::string](#) [table\\_name\\_](#)  
*Name of the blueprint of the currently placed entity.*
- [tdt::real](#) [half\\_height\\_](#)  
*Used to correctly place the newly created entity on the ground.*
- [bool](#) [placing\\_structure\\_](#)  
*Determines if a structure (wall, building etc.) is being placed, this will make the dummy entity to snap to the grid nodes.*
- [tdt::uint](#) [structure\\_radius\\_](#)  
*Radius of the placed structure (if it's covering more than one node).*
- [Ogre::SceneManager](#) & [mgr\\_](#)  
*Scene manager used to hold the dummy node and manipulate models.*
- [Ogre::Entity](#) \* [ent\\_](#)  
*Entity representing the mesh of the placed object.*
- [tdt::uint](#) [price\\_](#)  
*Price of the currently placed entity.*
- [bool](#) [can\\_place\\_when\\_game\\_paused\\_](#) {false}  
*Used for debugging, allows to place spawners when the game is paused and this their production is halted.*

### 6.20.1 Detailed Description

Class allowing the player/developer to place entities on the ground.

(Mainly used by the [EntityCreator](#) class, but can be invoked by "game.place\_entity(TABLE)" where TABLE is the name of the desired entity's blueprint)

Definition at line 15 of file EntityPlacer.hpp.

### 6.20.2 Constructor & Destructor Documentation

#### 6.20.2.1 EntityPlacer::EntityPlacer ( EntitySystem & ents, GridSystem & grid, Ogre::SceneManager & mgr )

Constructor.

##### Parameters

<i>Reference</i>	to the game's entity system.
<i>Reference</i>	to the game's grid system, used for node snapping when placing structures (i.e. walls, buildings...).
<i>Scene</i>	manager that will hold the dummy node.

Definition at line 8 of file EntityPlacer.cpp.

#### 6.20.2.2 EntityPlacer::~EntityPlacer ( )

Destructor.

Definition at line 15 of file EntityPlacer.cpp.

### 6.20.3 Member Function Documentation

#### 6.20.3.1 bool EntityPlacer::can\_place\_when\_game\_paused ( ) const

Returns true if the placer can place entities when the game is paused, false otherwise.

Definition at line 154 of file EntityPlacer.cpp.

#### 6.20.3.2 bool EntityPlacer::is\_visible ( ) const

Returns true if the placer is currently active (and thus the dummy entity visible), false otherwise.

Definition at line 144 of file EntityPlacer.cpp.

#### 6.20.3.3 tdt::uint EntityPlacer::place ( )

Creates a new entity from the blueprint table at the mouse cursor's current position and informs the developer in the developer console.

Definition at line 96 of file EntityPlacer.cpp.

#### 6.20.3.4 void EntityPlacer::set\_current\_entity\_table ( const std::string & table\_name, bool use\_price = false )

Checks if a given entity blueprint contains a graphics component and if so, sets it as the currently placed entity by creating a dummy entity following the cursor.

##### Parameters

<i>Name</i>	of the blueprint table (describing the placed entity).
<i>If</i>	true, the cost of the entity will be subtracted from the player's gold and the entity won't be placed if the player does not have sufficient funds.

Definition at line 22 of file EntityPlacer.cpp.

#### 6.20.3.5 void EntityPlacer::set\_visible ( bool on\_off )

Sets the visibility status of the placer (and it's dummy entity).

## Parameters

<i>The</i>	new visibility status.
------------	------------------------

Definition at line 135 of file EntityPlacer.cpp.

#### 6.20.3.6 void EntityPlacer::toggle\_placing\_when\_game\_paused ( )

Toggles (=negates) the flag for entity placing while the game is paused.

Definition at line 149 of file EntityPlacer.cpp.

#### 6.20.3.7 void EntityPlacer::update\_position ( const Ogre::Vector3 & pos )

Called every frame when the entity placer is active, adjusts the dummy entity's position to the cursor's.

## Parameters

<i>Position</i>	of the mouse cursor (recieved from <a href="#">Game::get_mouse_click_position</a> ).
-----------------	--

Definition at line 73 of file EntityPlacer.cpp.

### 6.20.4 Member Data Documentation

#### 6.20.4.1 bool EntityPlacer::can\_place\_when\_game\_paused\_ {false} [private]

Used for debugging, allows to place spawners when the game is paused and this their production is halted.

Definition at line 150 of file EntityPlacer.hpp.

#### 6.20.4.2 Ogre::Vector3 EntityPlacer::curr\_position\_ [private]

Current position of the dummy entity (used to avoid the need to pass the position again on the actual placement cal).

Definition at line 95 of file EntityPlacer.hpp.

#### 6.20.4.3 Ogre::Entity\* EntityPlacer::ent\_ [private]

Entity representing the mesh of the placed object.

Definition at line 139 of file EntityPlacer.hpp.

#### 6.20.4.4 EntitySystem& EntityPlacer::entities\_ [private]

Reference to the game's entity system.

(Used to create entities as well as checking the loaded tables.)

Definition at line 83 of file EntityPlacer.hpp.

#### 6.20.4.5 GridSystem& EntityPlacer::grid\_ [private]

Reference to the game's grid system.

(Used for node snapping when placing structures).

Definition at line 89 of file EntityPlacer.hpp.

#### 6.20.4.6 tdt::real EntityPlacer::half\_height\_ [private]

Used to correctly place the newly created entity on the ground.

(The mouse cursor position is at 0 Y coordinate, but the entity might need to be higher due to it's central point not being at the bottom of the model.)

Definition at line 118 of file EntityPlacer.hpp.

#### 6.20.4.7 Ogre::SceneManager& EntityPlacer::mgr\_ [private]

Scene manager used to hold the dummy node and manipulate models.

Definition at line 134 of file EntityPlacer.hpp.

#### 6.20.4.8 Ogre::SceneNode\* EntityPlacer::placing\_node\_ [private]

Pointer to the scene node the dummy entity is attached to.

Definition at line 100 of file EntityPlacer.hpp.

#### 6.20.4.9 bool EntityPlacer::placing\_structure\_ [private]

Determines if a structure (wall, building etc.) is being placed, this will make the dummy entity to snap to the grid nodes.

Definition at line 124 of file EntityPlacer.hpp.

#### 6.20.4.10 tdt::uint EntityPlacer::price\_ [private]

Price of the currently placed entity.

Definition at line 144 of file EntityPlacer.hpp.

6.20.4.11 `tdt::uint EntityPlacer::structure_radius_` `[private]`

Radius of the placed structure (if it's covering more than one node).

Definition at line 129 of file EntityPlacer.hpp.

6.20.4.12 `std::string EntityPlacer::table_name_` `[private]`

Name of the blueprint of the currently placed entity.

Definition at line 110 of file EntityPlacer.hpp.

6.20.4.13 `bool EntityPlacer::visible_` `[private]`

The placer's visibility (and active) status.

Definition at line 105 of file EntityPlacer.hpp.

The documentation for this class was generated from the following files:

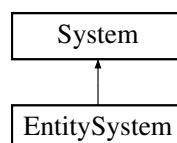
- tools/EntityPlacer.hpp
- tools/EntityPlacer.cpp

## 6.21 EntitySystem Class Reference

The [EntitySystem](#) class handles everything related to entities, like addition and removal of components, testing if an entity has a component or retrieval of components belonging to particular entities.

```
#include <EntitySystem.hpp>
```

Inheritance diagram for EntitySystem:





## Public Member Functions

- [EntitySystem](#) (Ogre::SceneManager &)  
*Constructor.*
- [~EntitySystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Checks for entities with no components and if any are found, deletes them.*
- tdt::uint [get\\_new\\_id](#) ()  
*Returns first available entity id.*
- void [cleanup](#) ()  
*Removes all entities that have no components and individual components marked for deletion from their entities (this is used so that the Lua code does not delete an entity/a component from a container while C++ iterates over it).*
- tdt::uint [create\\_entity](#) (const std::string &="", const Ogre::Vector3 &=Ogre::Vector3{0.f, 0.f, 0.f})  
*Creates a new entity from a blueprint.*
- const std::map< tdt::uint, std::bitset< Component::count > > & [get\\_component\\_list](#) () const  
*Breif: Returns const reference to the component list, so that it can be used to iterate over all entities.*
- template<typename COMP >  
bool [has\\_component](#) (tdt::uint id)  
*Tests whether a given entity has a component specialized by the template argument.*
- bool [has\\_component](#) (tdt::uint, tdt::uint) const  
*Tests whether a given entity has a component of a given type (used from Lua as it cannot use templates).*
- template<typename COMP >  
COMP \* [get\\_component](#) (tdt::uint id)  
*Returns a bool-component pointer pair, in which the first bool member determines if the component was found and the second is a pointer to the component.*
- template<typename COMP >  
void [set\\_component](#) (tdt::uint id, COMP comp)  
*Changes a component (type specified by template argument) of and entity or assigns a new component it that entity didn't have it.*
- template<typename COMP >  
std::map< tdt::uint, COMP > & [get\\_component\\_container](#) ()  
*Returns the map associated with the component specified by the template argument.*
- template<typename COMP >  
void [add\\_component](#) (tdt::uint id)  
*Adds a components to the given enetity using it's default constructor (all values have to be set afterwards).*
- void [add\\_component](#) (tdt::uint, int)  
*Allows to add a component based on it's ID.*
- template<typename COMP >  
void [delete\\_component](#) (tdt::uint id)  
*Marks a component (specified by template argument) for given entity for deletion.*
- void [delete\\_component](#) (tdt::uint, int)  
*Allows to enqueue a component for deletion based on it's ID.*
- void [register\\_entity](#) (const std::string &)  
*Registers an entity that has been loaded from a Lua script.*
- std::set< std::string > & [get\\_registered\\_entities](#) ()  
*Returns a reference to the set containing all entity tables registered during the game's runtime.*
- bool [exists](#) (tdt::uint) const  
*Checks if a given entity exists and returns true if it does, false otherwise.*
- Ogre::SceneManager & [get\\_scene\\_manager](#) ()  
*Returns a reference to the scene manager all entities of this system are attached to (if they have a graphics component).*
- void [delete\\_entities](#) ()

*Deletes all entities int the game, used before loading a new game.*

- `template<>`  
`std::map< tdt::uint, PhysicsComponent > & get_component_container ()`

*Specializations of the [EntitySystem::get\\_component\\_container](#) method.*

- `template<>`  
`std::map< tdt::uint, HealthComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, AIComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, GraphicsComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, MovementComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, CombatComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, EventComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, InputComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, TimeComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ManaComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, SpellComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ProductionComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, GridNodeComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ProductComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, PathfindingComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, TaskComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, TaskHandlerComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, StructureComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, HomingComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, EventHandlerComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, DestructorComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, GoldComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, FactionComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, PriceComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, AlignComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, MineComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ManaCrystalComponent > & get_component_container ()`

- `template<>`  
`std::map< tdt::uint, OnHitComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ConstructorComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, TriggerComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, UpgradeComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, NotificationComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ExplosionComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, LimitedLifeSpanComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, NameComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, ExperienceValueComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, LightComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, CommandComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, CounterComponent > & get_component_container ()`
- `template<>`  
`std::map< tdt::uint, PortalComponent > & get_component_container ()`

## Public Attributes

- `std::string NO\_BLUEPRINT {"ERROR"}`  
*Used in helpers when no component exists and we still need to return the blueprint name (in this case the ERROR blueprint) by reference.*
- `std::array< std::string, 3 > FACTION\_NAME {"FRIENDLY", "ENEMY", "NEUTRAL"}`  
*Used in when translating the faction enum to a string in the [FactionHelper](#).*

## Private Types

- `typedef void(EntitySystem::* LoaderFuncPtr) (tdt::uint, const std::string &)`
- `typedef void(EntitySystem::* AdderFuncPtr) (tdt::uint)`
- `typedef void(EntitySystem::* DeleterFuncPtr) (tdt::uint)`
- `typedef void(EntitySystem::* ImmediateDeleterFuncPtr) (tdt::uint)`

## Private Member Functions

- `template<typename COMP >`  
`void load\_component (tdt::uint id, const std::string &table_name)`  
*Loads a component from a Lua script.*
- `void destroy\_entity (tdt::uint)`  
*Removes an entity from the system, thus killing/destroying it.*
- `template<typename COMP >`  
`void delete\_component\_now (tdt::uint id)`  
*Deletes a component.*



- [illegible]

## Private Attributes

- `std::map< tdt::uint, std::bitset< Component::count > > entities_`  
*Contains bitsets describing component availability.*
- `std::vector< tdt::uint > to_be_destroyed_`  
*Used to mark components or entire entities for removal.*
- `std::vector< std::pair< tdt::uint, int > > components_to_be_removed_`
- `std::map< tdt::uint, PhysicsComponent > physics_ {}`  
*Contain components specified by the entity ID.*
- `std::map< tdt::uint, HealthComponent > health {}`

- `std::map< tdt::uint, AIComponent > ai_ {}`
- `std::map< tdt::uint, GraphicsComponent > graphics_ {}`
- `std::map< tdt::uint, MovementComponent > movement_ {}`
- `std::map< tdt::uint, CombatComponent > combat_ {}`
- `std::map< tdt::uint, EventComponent > event_ {}`
- `std::map< tdt::uint, InputComponent > input_ {}`
- `std::map< tdt::uint, TimeComponent > time_ {}`
- `std::map< tdt::uint, ManaComponent > mana_ {}`
- `std::map< tdt::uint, SpellComponent > spell_ {}`
- `std::map< tdt::uint, ProductionComponent > production_ {}`
- `std::map< tdt::uint, GridNodeComponent > grid_node_ {}`
- `std::map< tdt::uint, ProductComponent > product_ {}`
- `std::map< tdt::uint, PathfindingComponent > pathfinding_ {}`
- `std::map< tdt::uint, TaskComponent > task_ {}`
- `std::map< tdt::uint, TaskHandlerComponent > task_handler_ {}`
- `std::map< tdt::uint, StructureComponent > structure_ {}`
- `std::map< tdt::uint, HomingComponent > homing_ {}`
- `std::map< tdt::uint, EventHandlerComponent > event_handler_ {}`
- `std::map< tdt::uint, DestructorComponent > destructor_ {}`
- `std::map< tdt::uint, GoldComponent > gold_ {}`
- `std::map< tdt::uint, FactionComponent > faction_ {}`
- `std::map< tdt::uint, PriceComponent > price_ {}`
- `std::map< tdt::uint, AlignComponent > align_ {}`
- `std::map< tdt::uint, MineComponent > mine_ {}`
- `std::map< tdt::uint, ManaCrystalComponent > mana_crystal_ {}`
- `std::map< tdt::uint, OnHitComponent > on_hit_ {}`
- `std::map< tdt::uint, ConstructorComponent > constructor_ {}`
- `std::map< tdt::uint, TriggerComponent > trigger_ {}`
- `std::map< tdt::uint, UpgradeComponent > upgrade_ {}`
- `std::map< tdt::uint, NotificationComponent > notification_ {}`
- `std::map< tdt::uint, ExplosionComponent > explosion_ {}`
- `std::map< tdt::uint, LimitedLifeSpanComponent > limited_life_span_ {}`
- `std::map< tdt::uint, NameComponent > name_ {}`
- `std::map< tdt::uint, ExperienceValueComponent > exp_value_ {}`
- `std::map< tdt::uint, LightComponent > light_ {}`
- `std::map< tdt::uint, CommandComponent > command_ {}`
- `std::map< tdt::uint, CounterComponent > counter_ {}`
- `std::map< tdt::uint, PortalComponent > portal_ {}`
- `Ogre::SceneManager & scene_`

*Reference to the game's scene manager used to create nodes and entities.*

- `std::set< std::string > entity\_register\_`

*Contains the names of all loaded entity tables.*

- `std::array< LoaderFuncPtr, Component::count > loaders\_ {}`

*These arrays contain pointers to the component management methods for easier use when Lua interacts with C++, since Lua doesn't know anything about C++ types and templates.*

- `std::array< AdderFuncPtr, Component::count > adders\_ {}`
- `std::array< DeleterFuncPtr, Component::count > deleters\_ {}`
- `std::array< ImmediateDeleterFuncPtr, Component::count > immediate\_deleters\_ {}`
- `tdt::uint curr\_id\_ {}`

*Keeps track of the highest ID given to an entity.*

- `std::vector< tdt::uint > constructors\_to\_be\_called\_ {}`

*Entities that should have their constructors called on the next update (freshly created), this will allow the creator of the entity to set it's position if he could not pass that position to the create\_entity function.*

## Friends

- class `util::EntityDestroyer`

### 6.21.1 Detailed Description

The `EntitySystem` class handles everything related to entities, like addition and removal of components, testing if an entity has a component or retrieval of components belonging to particular entities.

Definition at line 21 of file `EntitySystem.hpp`.

### 6.21.2 Constructor & Destructor Documentation

#### 6.21.2.1 `EntitySystem::EntitySystem ( Ogre::SceneManager & mgr )`

Constructor.

##### Parameters

<i>Reference</i>	to the game's scene manager used to create nodes and entities.
------------------	--

Definition at line 32 of file `EntitySystem.cpp`.

#### 6.21.2.2 `EntitySystem::~~EntitySystem ( )` `[default]`

Destructor.

### 6.21.3 Member Function Documentation

#### 6.21.3.1 `template<typename COMP > void EntitySystem::add_component ( tdt::uint id )` `[inline]`

Adds a components to the given enetity using it's default constructor (all values have to be set afterwards).

##### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 140 of file `EntitySystem.hpp`.

#### 6.21.3.2 `void EntitySystem::add_component ( tdt::uint ent_id, int comp_id )`

Allows to add a component based on it's ID.

## Parameters

<i>ID</i>	of the entity.
<i>ID</i>	of the component.

Definition at line 154 of file EntitySystem.cpp.

**6.21.3.3** `template<typename COMP > void EntitySystem::clean_up_component ( tdt::uint ) [inline], [private]`

Deletes all necessary data when destroying a component (like Ogre related objects, other entities, tasks etc.).

## Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 263 of file EntitySystem.hpp.

**6.21.3.4** `template<> void EntitySystem::clean_up_component ( tdt::uint id ) [inline], [private]`

Specializations of the [EntitySystem::clean\\_up\\_component](#) method.

Definition at line 1044 of file EntitySystem.hpp.

**6.21.3.5** `template<> void EntitySystem::clean_up_component ( tdt::uint id ) [inline], [private]`

Since this is probably called while iterating over the list of all entities, creating new entity would probably invalidate the iterator, so the gold pile that is supposed to be picked up is given the event component (which is then removed when handled).

Better to not make it maximal, as we would still prefer a close miner to pick it up (it will increase in time if the event is not handled).

Definition at line 1057 of file EntitySystem.hpp.

**6.21.3.6** `template<> void EntitySystem::clean_up_component ( tdt::uint id ) [inline], [private]`

There is a slight chance that an entity would get killed right after destroying a gold deposit but before it could handle the event, so in that case just mark that event as global, so other miners can pick it up.

Definition at line 1102 of file EntitySystem.hpp.

**6.21.3.7** `void EntitySystem::cleanup ( )`

Removes all entities that have no components and individual components marked for deletion from their entities (this is used so that the Lua code does not delete an entity/a component from a container while C++ iterates over it).

Remove entire entities. NOTE: Creating a new vector and swaping it with the `to_be_destroyed_` vector, because when a [TaskHandlerComponent](#) is deleted, new entities (the tasks) are added which might result in iterator invalidation.

Definition at line 62 of file EntitySystem.cpp.



**6.21.3.8** `tdt::uint EntitySystem::create_entity ( const std::string & table_name = " ", const Ogre::Vector3 & position = Ogre::Vector3{0.f, 0.f, 0.f} )`

Creates a new entity from a blueprint.

#### Parameters

<i>Name</i>	of the Lua table containing the entity blueprint.
<i>Optional</i>	position of the entity.

Definition at line 109 of file EntitySystem.cpp.

**6.21.3.9** `template<typename COMP > void EntitySystem::delete_component ( tdt::uint id ) [inline]`

Marks a component (specified by template argument) for given entity for deletion.

#### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 162 of file EntitySystem.hpp.

**6.21.3.10** `void EntitySystem::delete_component ( tdt::uint ent_id, int comp_id )`

Allows to enqueue a component for deletion based on it's ID.

#### Parameters

<i>ID</i>	of the entity.
<i>ID</i>	of the component.

Definition at line 160 of file EntitySystem.cpp.

**6.21.3.11** `template<typename COMP > void EntitySystem::delete_component_now ( tdt::uint id ) [inline], [private]`

Deletes a component.

#### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 235 of file EntitySystem.hpp.

**6.21.3.12** `void EntitySystem::delete_component_now ( tdt::uint ent_id, int comp_id )` `[private]`

Deletes a component.

#### Parameters

<i>ID</i>	of the entity.
<i>ID</i>	of the component.

Definition at line 166 of file EntitySystem.cpp.

**6.21.3.13** `void EntitySystem::delete_entities ( )`

Deletes all entities int the game, used before loading a new game.

Definition at line 188 of file EntitySystem.cpp.

**6.21.3.14** `void EntitySystem::destroy_entity ( tdt::uint id )` `[private]`

Removes an entity from the system, thus killing/destroying it.

#### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 144 of file EntitySystem.cpp.

**6.21.3.15** `bool EntitySystem::exists ( tdt::uint id ) const`

Checks if a given entity exists and returns true if it does, false otherwise.

#### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 182 of file EntitySystem.cpp.

**6.21.3.16** `template<typename COMP > COMP* EntitySystem::get_component ( tdt::uint id )` `[inline]`

Returns a bool-component pointer pair, in which the first bool member determines if the component was found and the second is a pointer to the component.

#### Parameters

<i>ID</i>	of the entity whose component we ask for.
-----------	---

Definition at line 100 of file EntitySystem.hpp.

**6.21.3.17** `template<typename COMP > std::map<tdt::uint, COMP>& EntitySystem::get_component_container ( )`

Returns the map associated with the component specified by the template argument.

**6.21.3.18** `template<> std::map<tdt::uint, PhysicsComponent>& EntitySystem::get_component_container ( )`  
`[inline]`

Specializations of the [EntitySystem::get\\_component\\_container](#) method.

Definition at line 360 of file EntitySystem.hpp.

**6.21.3.19** `const std::map< tdt::uint, std::bitset< Component::count > > & EntitySystem::get_component_list ( ) const`

Breif: Returns const reference to the component list, so that it can be used to iterate over all entities.

Definition at line 149 of file EntitySystem.cpp.

**6.21.3.20** `tdt::uint EntitySystem::get_new_id ( )`

Returns first available entity id.

Definition at line 44 of file EntitySystem.cpp.

**6.21.3.21** `std::set< std::string > & EntitySystem::get_registered_entities ( )`

Returns a reference to the set containing all entity tables registered during the game's runtime.

Definition at line 177 of file EntitySystem.cpp.

**6.21.3.22** `Ogre::SceneManager& EntitySystem::get_scene_manager ( ) [inline]`

Returns a reference to the scene manager all entities of this system are attached to (if they have a graphics component).

Definition at line 197 of file EntitySystem.hpp.

**6.21.3.23** `template<typename COMP > bool EntitySystem::has_component ( tdt::uint id ) [inline]`

Tests whether a given entity has a component specialized by the template argument.

#### Parameters

<i>ID</i>	of the entity being checked.
-----------	------------------------------

**Note**

VS2015RC does not let me use variadic templates with recursion to check multiple components for some reason, investigate!

Definition at line 80 of file EntitySystem.hpp.

#### 6.21.3.24 `bool EntitySystem::has_component ( tdt::uint id, tdt::uint comp ) const`

Tests whether a given entity has a component of a given type (used from Lua as it cannot use templates).

**Parameters**

<i>ID</i>	of the entity.
<i>Type</i>	of the component.

Definition at line 362 of file EntitySystem.cpp.

#### 6.21.3.25 `void EntitySystem::init_function_arrays ( ) [private]`

Initializes all arrays holding pointers to the component manipulating methods.

Definition at line 195 of file EntitySystem.cpp.

#### 6.21.3.26 `template<typename COMP > void EntitySystem::load_component ( tdt::uint id, const std::string & table_name ) [private]`

Loads a component from a Lua script.

**Parameters**

<i>ID</i>	of the entity.
<i>Name</i>	of the table containing the component.

#### 6.21.3.27 `template<> void EntitySystem::load_component ( tdt::uint id, const std::string & table_name ) [inline], [private]`

Specializations of the [EntitySystem::load\\_component](#) method.

**Note**

Following components can only be created manually and thus don't have load\_component specialization. [GridNodeComponent](#) (created by GridSystem::add\_node) [ProductComponent](#) (production id is assigned during runtime) [TaskComponent](#) (tasks are specified by their types and are added through the [TaskHelper](#))

Definition at line 607 of file EntitySystem.hpp.

**6.21.3.28 void EntitySystem::register\_entity ( const std::string & *table\_name* )**

Registers an entity that has been loaded from a Lua script.

(If it has been registered previously, the register ignores it.)

**Parameters**

<i>Name</i>	of the table containing the info about the entity.
-------------	--

Definition at line 172 of file EntitySystem.cpp.

**6.21.3.29 template<typename COMP > void EntitySystem::set\_component ( tdt::uint *id*, COMP *comp* ) [inline]**

Changes a component (type specified by template argument) of an entity or assigns a new component to that entity didn't have it.

**Parameters**

<i>ID</i>	of the entity.
<i>Component</i>	to be assigned.

Definition at line 116 of file EntitySystem.hpp.

**6.21.3.30 void EntitySystem::update ( tdt::real ) [override],[virtual]**

Checks for entities with no components and if any are found, deletes them.

**Parameters**

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 39 of file EntitySystem.cpp.

**6.21.4 Member Data Documentation****6.21.4.1 std::vector<tdt::uint> EntitySystem::constructors\_to\_be\_called\_ {} [private]**

Entities that should have their constructors called on the next update (freshly created), this will allow the creator of the entity to set its position if he could not pass that position to the create\_entity function.

Definition at line 353 of file EntitySystem.hpp.

#### 6.21.4.2 `tdt::uint EntitySystem::curr_id_ {}` `[private]`

Keeps track of the highest ID given to an entity.

Definition at line 345 of file EntitySystem.hpp.

#### 6.21.4.3 `std::map<tdt::uint, std::bitset<Component::count> > EntitySystem::entities_` `[private]`

Contains bitsets describing component availability.

Definition at line 269 of file EntitySystem.hpp.

#### 6.21.4.4 `std::set<std::string> EntitySystem::entity_register_` `[private]`

Contains the names of all loaded entity tables.

Definition at line 330 of file EntitySystem.hpp.

#### 6.21.4.5 `std::array<std::string, 3> EntitySystem::FACTION_NAME {"FRIENDLY", "ENEMY", "NEUTRAL"}`

Used in when translating the faction enum to a string in the [FactionHelper](#).

Definition at line 213 of file EntitySystem.hpp.

#### 6.21.4.6 `std::array<LoaderFuncPtr, Component::count> EntitySystem::loaders_ {}` `[private]`

These arrays contain pointers to the component management methods for easier use when Lua interacts with C++, since Lua doesn't know anything about C++ types and templates.

Definition at line 337 of file EntitySystem.hpp.

#### 6.21.4.7 `std::string EntitySystem::NO_BLUEPRINT {"ERROR"}`

Used in helpers when no component exists and we still need to return the blueprint name (in this case the ERROR blueprint) by reference.

Definition at line 208 of file EntitySystem.hpp.

#### 6.21.4.8 `std::map<tdt::uint, PhysicsComponent> EntitySystem::physics_ {}` `[private]`

Contain components specified by the entity ID.

Initialized here to avoid a long initializing list in the constructor.

Definition at line 281 of file EntitySystem.hpp.

#### 6.21.4.9 Ogre::SceneManager& EntitySystem::scene\_ [private]

Reference to the game's scene manager used to create nodes and entities.

Definition at line 325 of file EntitySystem.hpp.

#### 6.21.4.10 std::vector<tdt::uint> EntitySystem::to\_be\_destroyed\_ [private]

Used to mark components or entire entities for removal.

Definition at line 274 of file EntitySystem.hpp.

The documentation for this class was generated from the following files:

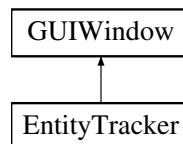
- systems/EntitySystem.hpp
- systems/EntitySystem.cpp

## 6.22 EntityTracker Class Reference

A window that monitors the stats of the currently selected entity and allows for it's upgrading once it has enough experience.

```
#include <EntityTracker.hpp>
```

Inheritance diagram for EntityTracker:



### Public Member Functions

- [EntityTracker](#) ()  
*Constructor.*
- [~EntityTracker](#) ()=default  
*Destructor.*
- void [set\\_tracked\\_entity](#) (tdt::uint, [EntitySystem](#) &)  
*Sets the new tracked entity and loads it's data.*
- tdt::uint [get\\_tracked\\_entity](#) () const  
*Returns the ID of the currently tracked entity.*
- void [update\\_tracking](#) (const std::string &, const std::string &)  
*Updates a single stat of the entity tracker.*
- void [clear](#) ()  
*Clears the entity tracker's window, that is sets all values to 0/0 and the id to UNKNOWN.*
- void [init\\_upgrade\\_but](#) ([EntitySystem](#) \*)  
*Adds a callback to the UPGRADE button that upgrades an entity.*
- void [show\\_upgrade\\_but](#) (bool)  
*Sets the visibility status of the UPGRADE button.*

## Protected Member Functions

- void `init_()` override  
*Initializes the window and sets all event subscribers.*

## Private Attributes

- `tdt::uint` `curr_tracked_entity_`  
*ID of the currently tracked entity.*
- `EntitySystem` \* `entities_`  
*Used for upgrading.*

## Additional Inherited Members

### 6.22.1 Detailed Description

A window that monitors the stats of the currently selected entity and allows for it's upgrading once it has enough experience.

Definition at line 11 of file `EntityTracker.hpp`.

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 `EntityTracker::EntityTracker ( )`

Constructor.

Definition at line 7 of file `EntityTracker.cpp`.

#### 6.22.2.2 `EntityTracker::~EntityTracker ( ) [default]`

Destructor.

### 6.22.3 Member Function Documentation

#### 6.22.3.1 `void EntityTracker::clear ( )`

Clears the entity tracker's window, that is sets all values to 0/0 and the id to UNKNOWN.

Definition at line 70 of file `EntityTracker.cpp`.

#### 6.22.3.2 `tdt::uint EntityTracker::get_tracked_entity ( ) const`

Returns the ID of the currently tracked entity.

Definition at line 59 of file `EntityTracker.cpp`.



### 6.22.3.3 void EntityTracker::init\_ ( ) [override],[protected],[virtual]

Initializes the window and sets all event subscribers.

Implements [GUIWindow](#).

Definition at line 138 of file EntityTracker.cpp.

### 6.22.3.4 void EntityTracker::init\_upgrade\_but ( EntitySystem \* ents )

Adds a callback to the UPGRADE button that upgrades an entity.

## Parameters

<i>Entity</i>	system containing entities that will be upgraded by this button.
---------------	--

Definition at line 87 of file EntityTracker.cpp.

#### 6.22.3.5 void EntityTracker::set\_tracked\_entity ( tdt::uint *id*, EntitySystem & *ents* )

Sets the new tracked entity and loads it's data.

## Parameters

<i>ID</i>	of the entity.
<i>EntitySystem</i>	that contains the entity.

Definition at line 11 of file EntityTracker.cpp.

#### 6.22.3.6 void EntityTracker::show\_upgrade\_butt ( bool *val* )

Sets the visibility status of the UPGRADE button.

## Parameters

<i>True</i>	for visible, false for invisible.
-------------	-----------------------------------

Definition at line 133 of file EntityTracker.cpp.

#### 6.22.3.7 void EntityTracker::update\_tracking ( const std::string & *label*, const std::string & *value* )

Updates a single stat of the entity tracker.

## Parameters

<i>Name</i>	of the stat label (e.g. "HP_LABEL", "GOLD_LABEL" etc).
<i>String</i>	with the new value.

## Note

The value should have the form "[CURRENT\_VALUE]/[MAX\_VALUE]".

Definition at line 64 of file EntityTracker.cpp.

## 6.22.4 Member Data Documentation

#### 6.22.4.1 tdt::uint EntityTracker::curr\_tracked\_entity\_ [private]

ID of the currently tracked entity.

Definition at line 72 of file EntityTracker.hpp.

#### 6.22.4.2 EntitySystem\* EntityTracker::entities\_ [private]

Used for upgrading.

Definition at line 77 of file EntityTracker.hpp.

The documentation for this class was generated from the following files:

- gui/EntityTracker.hpp
- gui/EntityTracker.cpp

## 6.23 EventComponent Struct Reference

Represents events that happen in the game, like gold seams dropping gold, curing an entity of poisoning or triggers from traps etc.

```
#include <Components.hpp>
```

### Public Member Functions

- **EventComponent** (EVENT\_TYPE ev=EVENT\_TYPE::NONE, tdt::uint t=Component::NO\_ENTITY, tdt::real r=0.f, bool a=true)
- **EventComponent** (const [EventComponent](#) &)=default
- **EventComponent** ([EventComponent](#) &&)=default
- [EventComponent](#) & **operator=** (const [EventComponent](#) &)=default
- [EventComponent](#) & **operator=** ([EventComponent](#) &&)=default

### Public Attributes

- EVENT\_TYPE **event\_type**
- tdt::uint **target**
- tdt::uint **handler**
- tdt::real **radius**
- bool **active**

### Static Public Attributes

- static constexpr int **type** = 6

### 6.23.1 Detailed Description

Represents events that happen in the game, like gold seams dropping gold, curing an entity of poisoning or triggers from traps etc.

Definition at line 183 of file Components.hpp.

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

- Components.hpp

## 6.24 EventHandlerComponent Struct Reference

Allows to cherry pick when it comes to event handling and handle only certain events.

```
#include <Components.hpp>
```

### Public Member Functions

- **EventHandlerComponent** (std::string &&h="ERROR")
- **EventHandlerComponent** (const [EventHandlerComponent](#) &)=default
- **EventHandlerComponent** ([EventHandlerComponent](#) &&)=default
- [EventHandlerComponent](#) & **operator=** (const [EventHandlerComponent](#) &)=default
- [EventHandlerComponent](#) & **operator=** ([EventHandlerComponent](#) &&)=default

### Public Attributes

- std::string **handler**
- std::bitset<(int) EVENT\_TYPE::COUNT > **possible\_events**

### Static Public Attributes

- static constexpr int **type** = 19

### 6.24.1 Detailed Description

Allows to cherry pick when it comes to event handling and handle only certain events.

(Also, only entities with this component will react to events.)

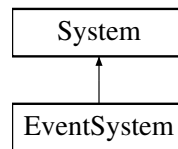
Definition at line 487 of file Components.hpp.

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

- Components.hpp

## 6.25 EventSystem Class Reference

Inheritance diagram for EventSystem:



### Public Member Functions

- [EventSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~EventSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Checks for handlers of all active events and if needed, handles them.*
- void [set\\_update\\_period](#) (tdt::real)  
*Sets the time it takes before an update is performed (in seconds).*
- tdt::real [get\\_update\\_period](#) () const  
*Returns the time it takes before an update is performed (in seconds).*
- void [set\\_update\\_time\\_multiplier](#) (tdt::real)  
*Sets the value by which the frame time is multiplied before it's added to the update timer.*
- tdt::real [get\\_update\\_time\\_multiplier](#) () const  
*Returns the value by which the frame time is multiplied before it's added to the update timer.*

### Private Member Functions

- bool [handle\\_event\\_](#) (tdt::uint, tdt::uint)  
*Handles a given event by a given entity that can handle it, returns true if the event will be destroyed after this call (single handler event), false if the event persist (multi handler event).*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system that has entities this system will manage.*
- tdt::real [update\\_period\\_](#)  
*Allow for less frequent updates, since per frame updates are quite unnecessary and resource wasting.*
- tdt::real [curr\\_update\\_time\\_](#)
- tdt::real [update\\_time\\_multiplier\\_](#)  
*Allows to speed up/slow down the update timer.*

#### 6.25.1 Detailed Description

Definition at line 7 of file EventSystem.hpp.

#### 6.25.2 Constructor & Destructor Documentation

##### 6.25.2.1 EventSystem::EventSystem ( EntitySystem & ents )

Constructor.

## Parameters

<i>Entity</i>	system that has entities this system will manage.
---------------	---

Definition at line 6 of file EventSystem.cpp.

#### 6.25.2.2 EventSystem::~EventSystem ( ) [default]

Destructor.

### 6.25.3 Member Function Documentation

#### 6.25.3.1 tdt::real EventSystem::get\_update\_period ( ) const

Returns the time it takes before an update is performed (in seconds).

Definition at line 63 of file EventSystem.cpp.

#### 6.25.3.2 tdt::real EventSystem::get\_update\_time\_multiplier ( ) const

Returns the value by which the frame time is multiplied before it's added to the update timer.

Definition at line 73 of file EventSystem.cpp.

#### 6.25.3.3 bool EventSystem::handle\_event\_ ( tdt::uint *handler*, tdt::uint *evt* ) [private]

Handles a given event by a given entity that can handle it, returns true if the event will be destroyed after this call (single handler event), false if the event persist (multi handler event).

## Parameters

<i>ID</i>	of the handler.
<i>ID</i>	of the event.

Definition at line 78 of file EventSystem.cpp.

#### 6.25.3.4 void EventSystem::set\_update\_period ( tdt::real *val* )

Sets the time it takes before an update is performed (in seconds).

## Parameters

<i>The</i>	new update time period.
------------	-------------------------

Definition at line 58 of file EventSystem.cpp.

#### 6.25.3.5 void EventSystem::set\_update\_time\_multiplier ( tdt::real *val* )

Sets the value by which the frame time is multiplied before it's added to the update timer.

##### Parameters

<i>The</i>	new multiplier value.
------------	-----------------------

Definition at line 68 of file EventSystem.cpp.

#### 6.25.3.6 void EventSystem::update ( tdt::real *delta* ) [override], [virtual]

Checks for handlers of all active events and if needed, handles them.

##### Parameters

<i>Time</i>	since last frame.
-------------	-------------------

Implements [System](#).

Definition at line 11 of file EventSystem.cpp.

## 6.25.4 Member Data Documentation

#### 6.25.4.1 EntitySystem& EventSystem::entities\_ [private]

Entity system that has entities this system will manage.

Definition at line 67 of file EventSystem.hpp.

#### 6.25.4.2 tdt::real EventSystem::update\_period\_ [private]

Allow for less frequent updates, since per frame updates are quite unnecessary and resource wasting.

Definition at line 73 of file EventSystem.hpp.

#### 6.25.4.3 tdt::real EventSystem::update\_time\_multiplier\_ [private]

Allows to speed up/slow down the update timer.

Definition at line 78 of file EventSystem.hpp.

The documentation for this class was generated from the following files:

- systems/EventSystem.hpp
- systems/EventSystem.cpp

## 6.26 lpp::Exception Class Reference

[Exception](#) class used to throw exception from the [Script](#) class.

```
#include <LppScript.hpp>
```

### Public Member Functions

- [Exception](#) (const std::string &msg="NO MSG", Script::state L=nullptr)  
*Constructor.*
- const char \* [what](#) () const  
*Returns the message of this exception.*
- const char \* [what\\_lua](#) () const  
*Returns the Lua error message if possible.*
- bool [has\\_lua\\_state](#) () const  
*Returns true if a Lua state is captured by this exception.*

### Private Attributes

- std::string [msg\\_](#)  
*Message the exception was called with.*
- Script::state [L\\_](#)  
*Pointer to the Lua state, used for stack manipulation (like lua error retrieval).*

#### 6.26.1 Detailed Description

[Exception](#) class used to throw exception from the [Script](#) class.

Definition at line 266 of file LppScript.hpp.

#### 6.26.2 Constructor & Destructor Documentation

6.26.2.1 `lpp::Exception::Exception ( const std::string & msg = "NO MSG", Script::state L = nullptr ) [inline]`

Constructor.

Parameters

<i>Message</i>	of the exception.
----------------	-------------------

Definition at line 273 of file LppScript.hpp.

#### 6.26.3 Member Function Documentation



#### 6.26.3.1 `bool lpp::Exception::has_lua_state ( ) const`

Returns true if a Lua state is captured by this exception.

Definition at line 128 of file LppScript.cpp.

#### 6.26.3.2 `const char * lpp::Exception::what ( ) const`

Returns the message of this exception.

[lpp::Exception](#) definitions:

Definition at line 114 of file LppScript.cpp.

#### 6.26.3.3 `const char * lpp::Exception::what_lua ( ) const`

Returns the Lua error message if possible.

Definition at line 119 of file LppScript.cpp.

### 6.26.4 Member Data Documentation

#### 6.26.4.1 `Script::state lpp::Exception::L_ [private]`

Pointer to the Lua state, used for stack manipulation (like lua error retrieval).

Definition at line 301 of file LppScript.hpp.

#### 6.26.4.2 `std::string lpp::Exception::msg_ [private]`

Message the exception was called with.

Definition at line 296 of file LppScript.hpp.

The documentation for this class was generated from the following files:

- lppscript/LppScript.hpp
- lppscript/LppScript.cpp

## 6.27 ExperienceValueComponent Struct Reference

The amount of experience the entity yields when killed.

```
#include <Components.hpp>
```

## Public Member Functions

- **ExperienceValueComponent** (tdt::uint v=0)
- **ExperienceValueComponent** (const [ExperienceValueComponent](#) &)=default
- **ExperienceValueComponent** ([ExperienceValueComponent](#) &&)=default
- [ExperienceValueComponent](#) & **operator=** (const [ExperienceValueComponent](#) &)=default
- [ExperienceValueComponent](#) & **operator=** ([ExperienceValueComponent](#) &&)=default

## Public Attributes

- tdt::uint **value**

## Static Public Attributes

- static constexpr int **type** = 35

### 6.27.1 Detailed Description

The amount of experience the entity yields when killed.

Definition at line 822 of file Components.hpp.

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

- Components.hpp

## 6.28 ExplosionComponent Struct Reference

[Component](#) used to create the visual effect of an explosion, the damage should be done in the explosion's constructor so that it's not applied on each frame.

```
#include <Components.hpp>
```

## Public Member Functions

- **ExplosionComponent** (tdt::real d=0.f, tdt::real rad=0.f)
- **ExplosionComponent** (const [ExplosionComponent](#) &)=default
- **ExplosionComponent** ([ExplosionComponent](#) &&)=default
- [ExplosionComponent](#) & **operator=** (const [ExplosionComponent](#) &)=default
- [ExplosionComponent](#) & **operator=** ([ExplosionComponent](#) &&)=default

## Public Attributes

- tdt::real **delta**
- tdt::real **max\_radius**
- tdt::real **curr\_radius**

### Static Public Attributes

- static constexpr int **type** = 32

#### 6.28.1 Detailed Description

[Component](#) used to create the visual effect of an explosion, the damage should be done in the explosion's constructor so that it's not applied on each frame.

Definition at line 761 of file Components.hpp.

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

- Components.hpp

## 6.29 FactionComponent Struct Reference

Represents the faction an entity that has this component if a member of.

```
#include <Components.hpp>
```

### Public Member Functions

- **FactionComponent** (FACTION f=FACTION::NEUTRAL)
- **FactionComponent** (const [FactionComponent](#) &)=default
- **FactionComponent** ([FactionComponent](#) &&)=default
- [FactionComponent](#) & **operator=** (const [FactionComponent](#) &)=default
- [FactionComponent](#) & **operator=** ([FactionComponent](#) &&)=default

### Public Attributes

- FACTION **faction**

### Static Public Attributes

- static constexpr int **type** = 22

#### 6.29.1 Detailed Description

Represents the faction an entity that has this component if a member of.

Definition at line 549 of file Components.hpp.

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

- Components.hpp

## 6.30 util::path\_type::FIRST\_PATH Struct Reference

Finds the first path by accepting the first path found.

```
#include <PathfindingAlgorithms.hpp>
```

### Static Public Member Functions

- static bool **return\_path** ()

#### 6.30.1 Detailed Description

Finds the first path by accepting the first path found.

Definition at line 154 of file PathfindingAlgorithms.hpp.

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

- tools/PathfindingAlgorithms.hpp

## 6.31 util::effect::FREEZE\_EFFECT Struct Reference

Freezes a given entity in place for a given time period.

```
#include <Effects.hpp>
```

### Public Member Functions

- [FREEZE\\_EFFECT](#) ([EntitySystem](#) &, tdt::real)  
*Constructor.*
- [~FREEZE\\_EFFECT](#) ()=default  
*Destructor.*
- void [operator\(\)](#) (tdt::uint)  
*Freezes a given entity in place.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing the entities this effect will be called on.*
- tdt::real [time\\_](#)  
*The duration of the freeze.*

#### 6.31.1 Detailed Description

Freezes a given entity in place for a given time period.

(This effect stops movement but not any other action.)

Definition at line 133 of file Effects.hpp.

#### 6.31.2 Constructor & Destructor Documentation

##### 6.31.2.1 util::effect::FREEZE\_EFFECT::FREEZE\_EFFECT ( [EntitySystem](#) & *ents*, tdt::real *time* )

Constructor.

## Parameters

<i>Entity</i>	system containing the entities this effect will be called on.
<i>Duration</i>	of the freeze.

Definition at line 58 of file Effects.cpp.

#### 6.31.2.2 util::effect::FREEZE\_EFFECT::~~FREEZE\_EFFECT ( ) [default]

Destructor.

### 6.31.3 Member Function Documentation

#### 6.31.3.1 void util::effect::FREEZE\_EFFECT::operator() ( tdt::uint *id* )

Freezes a given entity in place.

## Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 62 of file Effects.cpp.

### 6.31.4 Member Data Documentation

#### 6.31.4.1 EntitySystem& util::effect::FREEZE\_EFFECT::entities\_ [private]

Entity system containing the entities this effect will be called on.

Definition at line 159 of file Effects.hpp.

#### 6.31.4.2 tdt::real util::effect::FREEZE\_EFFECT::time\_ [private]

The duration of the freeze.

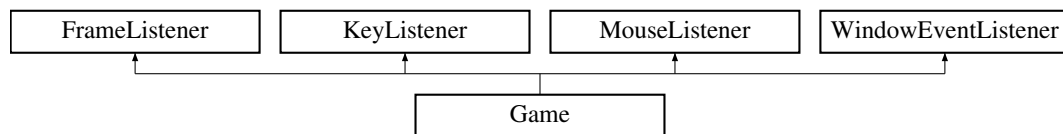
Definition at line 164 of file Effects.hpp.

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

- tools/Effects.hpp
- tools/Effects.cpp

## 6.32 Game Class Reference

Inheritance diagram for Game:



### Public Member Functions

- [Game](#) ()  
*Constructor.*
- [~Game](#) ()  
*Destructor.*
- void [run](#) ()  
*Starts the game.*
- void [update](#) (tdt::real)  
*Updates the game in one frame.*
- void [set\\_state](#) (GAME\_STATE)  
*Changes the game's state.*
- void [new\\_game](#) (tdt::uint, tdt::uint)  
*Creates a new game with the given dimensions.*
- void [create\\_empty\\_level](#) (tdt::uint, tdt::uint)  
*Creates an empty level with the given dimensions.*
- void [reset\\_camera](#) ()  
*Resets the main camera's position and orientation to it's original state.*
- void [reset\\_unlocks](#) ()  
*Restores the unlocks to their initial state.*
- void [set\\_throne\\_id](#) (tdt::uint)  
*Sets the ID of the entity that represents the Dungeon Throne.*
- tdt::uint [get\\_throne\\_id](#) () const  
*Returns the ID of the entity that represents the Dungeon Throne.*

### Protected Member Functions

- bool [frameRenderingQueued](#) (const Ogre::FrameEvent &) override  
*Called when the previous frame is queued for rendering, used for game updates.*
- bool [keyPressed](#) (const OIS::KeyEvent &) override  
*Called when a key is pressed.*
- bool [keyReleased](#) (const OIS::KeyEvent &) override  
*Called when a key is released.*
- bool [mouseMoved](#) (const OIS::MouseEvent &) override  
*Called when the mouse is moved.*
- bool [mousePressed](#) (const OIS::MouseEvent &, OIS::MouseButtonID) override  
*Called when a mouse button is pressed.*
- bool [mouseReleased](#) (const OIS::MouseEvent &, OIS::MouseButtonID) override  
*Called when a mouse button is released.*
- void [windowResized](#) (Ogre::RenderWindow \*rw) override  
*Called when the window is resized.*
- void [windowClosed](#) (Ogre::RenderWindow \*rw) override  
*Called when the window is closed.*

## Private Member Functions

- void [ogre\\_init](#) ()  
*Init methods.*
- void [ois\\_init](#) ()
- void [cegui\\_init](#) ()
- CEGUI::MouseButton [ois\\_to\\_cegui](#) (OIS::MouseButtonID)  
*Converts an OIS key code to CEGUI key code.*
- void [toggle\\_camera\\_free\\_mode](#) ()  
*Toggles the free camera movement mode.*
- std::pair< bool, Ogre::Vector3 > [get\\_mouse\\_click\\_position](#) (const OIS::MouseEvent &) const  
*Returns a pair consisting of the location of the point where the player clicked on the ground plane and a boolean indicating if the click was indeed on the ground plane.*

## Private Attributes

- GAME\_STATE [state\\_](#)  
*Current game state.*
- std::unique\_ptr< Ogre::Root > [root\\_](#)  
*Pointers to Ogre3D objects.*
- Ogre::SceneManager \* [scene\\_mgr\\_](#)
- Ogre::RenderWindow \* [window\\_](#)
- Ogre::Viewport \* [main\\_view\\_](#)
- Ogre::Light \* [main\\_light\\_](#)
- OIS::InputManager \* [input\\_](#)
- OIS::Keyboard \* [keyboard\\_](#)
- OIS::Mouse \* [mouse\\_](#)
- std::unique\_ptr< [Camera](#) > [main\\_cam\\_](#)  
*[Camera](#) rendering to the window.*
- std::unique\_ptr< [EntitySystem](#) > [entity\\_system\\_](#) {nullptr}  
*Unique pointers to systems (sadly all systems require the [EntitySystem](#), which requires Ogre::SceneManager, so the all have to be instantiated after the [ogre\\_init](#) method call and thus cannot be references).*
- std::unique\_ptr< [HealthSystem](#) > [health\\_system\\_](#) {nullptr}
- std::unique\_ptr< [MovementSystem](#) > [movement\\_system\\_](#) {nullptr}
- std::unique\_ptr< [AISystem](#) > [ai\\_system\\_](#) {nullptr}
- std::unique\_ptr< [InputSystem](#) > [input\\_system\\_](#) {nullptr}
- std::unique\_ptr< [GridSystem](#) > [grid\\_system\\_](#) {nullptr}
- std::unique\_ptr< [TaskSystem](#) > [task\\_system\\_](#) {nullptr}
- std::unique\_ptr< [CombatSystem](#) > [combat\\_system\\_](#) {nullptr}
- std::unique\_ptr< [ProductionSystem](#) > [production\\_system\\_](#) {nullptr}
- std::unique\_ptr< [TimeSystem](#) > [time\\_system\\_](#) {nullptr}
- std::unique\_ptr< [EventSystem](#) > [event\\_system\\_](#) {nullptr}
- std::unique\_ptr< [GraphicsSystem](#) > [graphics\\_system\\_](#) {nullptr}
- std::unique\_ptr< [TriggerSystem](#) > [trigger\\_system\\_](#) {nullptr}
- std::unique\_ptr< [ManaSpellSystem](#) > [mana\\_spell\\_system\\_](#) {nullptr}
- std::unique\_ptr< [WaveSystem](#) > [wave\\_system\\_](#) {nullptr}
- std::unique\_ptr< [GameSerializer](#) > [game\\_serializer\\_](#) {nullptr}  
*Used to save the game.*
- std::vector< [System](#) \* > [systems\\_](#) {}  
*Vector of all systems used for updating the game's logic.*
- CEGUI::OgreRenderer \* [renderer\\_](#)  
*CEGUI renderer.*

- `std::unique_ptr< EntityPlacer > placer\_`  
*Allows to spawn entities with the mouse ingame.*
- `std::unique_ptr< Ogre::Plane > ground\_`  
*The game's ground represented by a plane, used for ray intersections.*
- `Ogre::Entity * ground\_entity\_`  
*Entity holding the ground plane.*
- `std::unique_ptr< SelectionBox > selection\_box\_`  
*Selection box used to select multiple entities at once.*
- `std::unique_ptr< EntityCreator > entity\_creator\_`  
*A gui window that allows to place entities selected from a menu.*
- `Ogre::Vector2 mouse\_position\_`  
*Saved position of the mouse cursor (2D).*
- `std::unique_ptr< level\_generators::LevelGenerator > level\_generator\_`  
*Used to create new levels (that is, to distribute walls and gold deposits).*
- `std::unique_ptr< Spellcaster > spell\_caster\_`  
*Allows the player to cast spells of different types (positional, targeted etc.).*
- `tdt::uint throne\_id\_`  
*ID of the entity representing the Dungeon Throne, losing which ends the game.*

## Friends

- class **GameSerializer**
- class **LuaInterface**
- class **GUI**
- void **action::QUICK\_LOAD** ()
- void **action::QUICK\_SAVE** ()
- void **action::RESET\_CAMERA** ()

## 6.32.1 Detailed Description

Definition at line 40 of file `Game.hpp`.

## 6.32.2 Constructor & Destructor Documentation

### 6.32.2.1 `Game::Game ( )`

Constructor.

Definition at line 30 of file `Game.cpp`.

### 6.32.2.2 `Game::~~Game ( )`

Destructor.

Definition at line 104 of file `Game.cpp`.

## 6.32.3 Member Function Documentation

### 6.32.3.1 `void Game::create_empty_level ( tdt::uint width, tdt::uint height )`

Creates an empty level with the given dimensions.



## Parameters

<i>Width</i>	of the level.
<i>Height</i>	of the level.

Definition at line 186 of file Game.cpp.

**6.32.3.2** `bool Game::frameRenderingQueued ( const Ogre::FrameEvent & event ) [override], [protected]`

Called when the previous frame is queued for rendering, used for game updates.

Definition at line 264 of file Game.cpp.

**6.32.3.3** `std::pair< bool, Ogre::Vector3 > Game::get_mouse_click_position ( const OIS::MouseEvent & event ) const [private]`

Returns a pair consisting of the location of the point where the player clicked on the ground plane and a boolean indicating if the click was indeed on the ground plane.

Definition at line 631 of file Game.cpp.

**6.32.3.4** `tdt::uint Game::get_throne_id ( ) const`

Returns the ID of the entity that represents the Dungeon Throne.

Definition at line 259 of file Game.cpp.

**6.32.3.5** `bool Game::keyPressed ( const OIS::KeyEvent & event ) [override], [protected]`

Called when a key is pressed.

Definition at line 278 of file Game.cpp.

**6.32.3.6** `bool Game::keyReleased ( const OIS::KeyEvent & event ) [override], [protected]`

Called when a key is released.

This will make sure global and targeted spells go off even when a hotkey is used.

Definition at line 318 of file Game.cpp.

**6.32.3.7** `bool Game::mouseMoved ( const OIS::MouseEvent & event ) [override], [protected]`

Called when the mouse is moved.

Definition at line 348 of file Game.cpp.

**6.32.3.8** `bool Game::mousePressed ( const OIS::MouseEvent & event, OIS::MouseButtonID id )` `[override]`,  
`[protected]`

Called when a mouse button is pressed.

Note: If the spell is targeted and not one entity is selected, the spell shall be casted on mouse button release that makes sufficient selection.

Definition at line 384 of file Game.cpp.

**6.32.3.9** `bool Game::mouseReleased ( const OIS::MouseEvent & event, OIS::MouseButtonID id )` `[override]`,  
`[protected]`

Called when a mouse button is released.

Note: Selected the spell without a target, so when we select one target, apply the effect on him. This will also cause global spells to be casted immediately after clicking the button.

Definition at line 435 of file Game.cpp.

**6.32.3.10** `void Game::new_game ( tdt::uint width, tdt::uint height )`

Creates a new game with the given dimensions.

Parameters

<i>Width</i>	of the level.
<i>Height</i>	of the level.

Definition at line 172 of file Game.cpp.

**6.32.3.11** `void Game::ogre_init ( )` `[private]`

Init methods.

Definition at line 503 of file Game.cpp.

**6.32.3.12** `CEGUI::MouseButton Game::ois_to_cegui ( OIS::MouseButtonID id )` `[private]`

Converts an OIS key code to CEGUI key code.

Parameters

<i>OIS</i>	key code.
------------	-----------

Definition at line 611 of file Game.cpp.

**6.32.3.13** void Game::reset\_camera ( )

Resets the main camera's position and orientation to it's original state.

Definition at line 231 of file Game.cpp.

**6.32.3.14** void Game::reset\_unlocks ( )

Restores the unlocks to their initial state.

Definition at line 236 of file Game.cpp.

**6.32.3.15** void Game::run ( )

Starts the game.

Definition at line 110 of file Game.cpp.

**6.32.3.16** void Game::set\_state ( GAME\_STATE *state* )

Changes the game's state.

**Parameters**

<i>The</i>	new state.
------------	------------

Definition at line 154 of file Game.cpp.

**6.32.3.17** void Game::set\_throne\_id ( tdt::uint *id* )

Sets the ID of the entity that represents the Dungeon Throne.

**Parameters**

<i>The</i>	new ID.
------------	---------

Definition at line 254 of file Game.cpp.

**6.32.3.18** void Game::toggle\_camera\_free\_mode ( ) [private]

Toggles the free camera movement mode.

Definition at line 626 of file Game.cpp.

6.32.3.19 void Game::update ( tdt::real *delta* )

Updates the game in one frame.

## Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Definition at line 118 of file Game.cpp.

**6.32.3.20** `void Game::windowClosed ( Ogre::RenderWindow * rw )` `[override], [protected]`

Called when the window is closed.

Definition at line 485 of file Game.cpp.

**6.32.3.21** `void Game::windowResized ( Ogre::RenderWindow * rw )` `[override], [protected]`

Called when the window is resized.

Definition at line 471 of file Game.cpp.

## 6.32.4 Member Data Documentation

**6.32.4.1** `std::unique_ptr<EntityCreator> Game::entity_creator_` `[private]`

A gui window that allows to place entities selected from a menu.

(TODO: Allow to create/modify entities.)

Definition at line 272 of file Game.hpp.

**6.32.4.2** `std::unique_ptr<EntitySystem> Game::entity_system_ {nullptr}` `[private]`

Unique pointers to systems (sadly all systems require the [EntitySystem](#), which requires `Ogre::SceneManager`, so the all have to be instantiated after the `ogre_init` method call and thus cannot be references).

Definition at line 199 of file Game.hpp.

**6.32.4.3** `std::unique_ptr<GameSerializer> Game::game_serializer_ {nullptr}` `[private]`

Used to save the game.

Definition at line 218 of file Game.hpp.

**6.32.4.4** `std::unique_ptr<Ogre::Plane> Game::ground_` `[private]`

The game's ground represented by a plane, used for ray intersections.

Definition at line 244 of file Game.hpp.

#### 6.32.4.5 `Ogre::Entity*` `Game::ground_entity_` [private]

Entity holding the ground plane.

Used for easier plane switching.

Definition at line 249 of file Game.hpp.

#### 6.32.4.6 `std::unique_ptr<level_generators::LevelGenerator>` `Game::level_generator_` [private]

Used to create new levels (that is, to distribute walls and gold deposits).

Definition at line 283 of file Game.hpp.

#### 6.32.4.7 `std::unique_ptr<Camera>` `Game::main_cam_` [private]

[Camera](#) rendering to the window.

Definition at line 192 of file Game.hpp.

#### 6.32.4.8 `Ogre::Vector2` `Game::mouse_position_` [private]

Saved position of the mouse cursor (2D).

Definition at line 277 of file Game.hpp.

#### 6.32.4.9 `std::unique_ptr<EntityPlacer>` `Game::placer_` [private]

Allows to spawn entities with the mouse ingame.

Definition at line 239 of file Game.hpp.

#### 6.32.4.10 `CEGUI::OgreRenderer*` `Game::renderer_` [private]

CEGUI renderer.

Definition at line 228 of file Game.hpp.

#### 6.32.4.11 `std::unique_ptr<Ogre::Root>` `Game::root_` [private]

Pointers to Ogre3D objects.

Definition at line 179 of file Game.hpp.

6.32.4.12 `std::unique_ptr<SelectionBox> Game::selection_box_ [private]`

Selection box used to select multiple entities at once.

Definition at line 266 of file Game.hpp.

6.32.4.13 `std::unique_ptr<Spellcaster> Game::spell_caster_ [private]`

Allows the player to cast spells of different types (positional, targeted etc.).

Definition at line 288 of file Game.hpp.

6.32.4.14 `GAME_STATE Game::state_ [private]`

Current game state.

Definition at line 174 of file Game.hpp.

6.32.4.15 `std::vector<System*> Game::systems_ {} [private]`

Vector of all systems used for updating the game's logic.

Definition at line 223 of file Game.hpp.

6.32.4.16 `tdt::uint Game::throne_id_ [private]`

ID of the entity representing the Dungeon Throne, losing which ends the game.

Definition at line 294 of file Game.hpp.

The documentation for this class was generated from the following files:

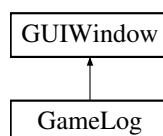
- Game.hpp
- Game.cpp

## 6.33 GameLog Class Reference

Class representing the log window used to show messages to the player.

```
#include <GameLog.hpp>
```

Inheritance diagram for GameLog:



## Public Member Functions

- [GameLog](#) ()  
*Constructor.*
- [~GameLog](#) ()=default  
*Destructor.*
- void [clear](#) ()  
*Clears the game's log by deleting all it's entries.*
- void [print](#) (const std::string &)  
*Prints a string to the game's log.*
- void [set\\_history](#) (tdt::uint)  
*Sets the amount of entries kept in the game's log.*
- tdt::uint [get\\_history](#) () const  
*Returns the amoung of entries kept in the game's log.*

## Protected Member Functions

- void [init\\_](#) () override  
*Initializes the game log (called by parent's init).*

## Private Attributes

- tdt::uint [log\\_history\\_](#)  
*Number of entires kept in the game log.*
- CEGUI::Listbox \* [log\\_](#)  
*Pointer to the log window for easy access (as it might get called quite often, this will avoid frequent lookups).*

## Additional Inherited Members

### 6.33.1 Detailed Description

Class representing the log window used to show messages to the player.

(No space for gold, enemies attacking etc.)

#### Note

For debug/technical etc messages, see the [Console](#) class.

Definition at line 15 of file GameLog.hpp.

### 6.33.2 Constructor & Destructor Documentation

#### 6.33.2.1 GameLog::GameLog ( )

Constructor.

Definition at line 4 of file GameLog.cpp.



#### 6.33.2.2 GameLog::~GameLog ( ) [default]

Destructor.

### 6.33.3 Member Function Documentation

#### 6.33.3.1 void GameLog::clear ( )

Clears the game's log by deleting all it's entries.

Definition at line 8 of file GameLog.cpp.

#### 6.33.3.2 tdt::uint GameLog::get\_history ( ) const

Returns the amoung of entries kept in the game's log.

Definition at line 36 of file GameLog.cpp.

#### 6.33.3.3 void GameLog::init\_ ( ) [override],[protected],[virtual]

Initializes the game log (called by parent's init).

Implements [GUIWindow](#).

Definition at line 41 of file GameLog.cpp.

#### 6.33.3.4 void GameLog::print ( const std::string & msg )

Prints a string to the game's log.

##### Parameters

<i>String</i>	to be printed.
---------------	----------------

Definition at line 13 of file GameLog.cpp.

#### 6.33.3.5 void GameLog::set\_history ( tdt::uint val )

Sets the amount of entries kept in the game's log.

##### Parameters

<i>The</i>	new log history.
------------	------------------

Definition at line 31 of file GameLog.cpp.

### 6.33.4 Member Data Documentation

#### 6.33.4.1 CEGUI::Listbox\* GameLog::log\_ [private]

Pointer to the log window for easy access (as it might get called quite often, this will avoid frequent lookups).

Definition at line 66 of file GameLog.hpp.

#### 6.33.4.2 tdt::uint GameLog::log\_history\_ [private]

Number of entires kept in the game log.

Definition at line 60 of file GameLog.hpp.

The documentation for this class was generated from the following files:

- gui/GameLog.hpp
- gui/GameLog.cpp

## 6.34 GameSerializer Class Reference

Class that is used to save (by using Lua code generation) and loading the game (by executing said code).

```
#include <GameSerializer.hpp>
```

### Public Member Functions

- [GameSerializer](#) ([EntitySystem](#) &)  
*Constructor.*
- [~GameSerializer](#) ()  
*Destructor.*
- void [save\\_game](#) ([Game](#) &, const std::string &="quick\_save")  
*Creates a Lua script that is to be used as a save file by serializing every entity into a sequence of commands that create this entity from scratch when executed.*
- void [load\\_game](#) ([Game](#) &, const std::string &="quick\_save")  
*Executes a given Lua script containing a serialized game, effectively restoring the state of that game.*

### Private Types

- typedef void(GameSerializer::\* **SerializerFuncPtr**) (tdt::uint, const std::string &)

## Private Member Functions

- [illegible]

- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)
- `template<>`  
void **save\_component** (tdt::uint id, const std::string &tbl\_name)

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system, used for component access.*
- [lpp::Script](#) & [script\\_](#)  
*Reference to the [lpp::Script](#) singleton for easy use.*
- `std::vector< std::pair< tdt::uint, tdt::uint > >` [task\\_pairs\\_](#)  
*Contains entity - task pairs that should be added in the `save_tasks` method.*
- `std::ofstream` [file\\_](#)  
*Main file stream (no need for `ifstream`, since loading is done through Lua).*
- `std::vector< std::string >` [save\\_entities\\_](#)  
*Auxiliary vectors that allows to place entity creation at the top (so no entity variables are nil when loading a game) and component definitions at the bottom.*
- `std::vector< std::string >` [save\\_components\\_](#)
- `std::array< SerializerFuncPtr, Component::count >` [serializers\\_](#)  
*Pointers to the different `save_component` instances allowing for easy runtime differencing between components.*

### 6.34.1 Detailed Description

Class that is used to save (by using Lua code generation) and loading the game (by executing said code).

Definition at line 22 of file GameSerializer.hpp.

### 6.34.2 Constructor & Destructor Documentation

#### 6.34.2.1 GameSerializer::GameSerializer ( EntitySystem & ents )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 17 of file GameSerializer.cpp.

#### 6.34.2.2 GameSerializer::~GameSerializer ( ) [inline]

Destructor.

Definition at line 35 of file GameSerializer.hpp.

### 6.34.3 Member Function Documentation

#### 6.34.3.1 void GameSerializer::load\_game ( Game & game, const std::string & fname = "quick\_save" )

Executes a given Lua script containing a serialized game, effectively restoring the state of that game.

Parameters

<i>Reference</i>	to the game object (currently used for console entries, but might be used more in the future).
<i>Name</i>	of the save file to load.

Definition at line 142 of file GameSerializer.cpp.

#### 6.34.3.2 template<typename COMP > void GameSerializer::save\_component ( tdt::uint , const std::string & ) [private]

Generates code that constructs a single component.

Parameters

<i>ID</i>	of the component to serialize (type specialized as template argument).
<i>Name</i>	of the variable already in the save file that holds the new ID.

#### 6.34.3.3 void GameSerializer::save\_game ( Game & game, const std::string & fname = "quick\_save" )

Creates a Lua script that is to be used as a save file by serializing every entity into a sequence of commands that create this entity from scratch when executed.

##### Parameters

<i>Reference</i>	to the <a href="#">Game</a> object (to be able to save all necessary data).
<i>Name</i>	of the save file.

Definition at line 64 of file GameSerializer.cpp.

#### 6.34.3.4 void GameSerializer::save\_tasks ( ) [private]

Adds commands to the save file that assign all tasks (has to be done last).

Definition at line 172 of file GameSerializer.cpp.

#### 6.34.3.5 std::string GameSerializer::save\_unlocks ( ) [private]

Returns a string containing commands that will restore the unlock system to it's current state.

Definition at line 213 of file GameSerializer.cpp.

#### 6.34.3.6 std::string GameSerializer::save\_wave\_system ( Game & game ) [private]

Returns a string containing commands that will restore the wave system to it's current state.

##### Parameters

<i>Reference</i>	to the game object that contains the wave system.
------------------	---

Definition at line 183 of file GameSerializer.cpp.

### 6.34.4 Member Data Documentation

#### 6.34.4.1 EntitySystem& GameSerializer::entities\_ [private]

Reference to the game's entity system, used for component access.

Definition at line 85 of file GameSerializer.hpp.

#### 6.34.4.2 std::ofstream GameSerializer::file\_ [private]

Main file stream (no need for ifstream, since loading is done through Lua).

Definition at line 100 of file GameSerializer.hpp.

#### 6.34.4.3 `std::vector<std::string> GameSerializer::save_entities_` [private]

Auxiliary vectors that allows to place entity creation at the top (so no entity variables are nil when loading a game) and component definitions at the bottom.

Definition at line 106 of file GameSerializer.hpp.

#### 6.34.4.4 `lpp::Script& GameSerializer::script_` [private]

Reference to the [lpp::Script](#) singleton for easy use.

Definition at line 90 of file GameSerializer.hpp.

#### 6.34.4.5 `std::array<SerializerFuncPtr, Component::count> GameSerializer::serializers_` [private]

Pointers to the different save\_component instances allowing for easy runtime differencing between components.

Definition at line 112 of file GameSerializer.hpp.

#### 6.34.4.6 `std::vector<std::pair<tdt::uint, tdt::uint> > GameSerializer::task_pairs_` [private]

Contains entity - task pairs that should be added in the save\_tasks method.

Definition at line 95 of file GameSerializer.hpp.

The documentation for this class was generated from the following files:

- tools/GameSerializer.hpp
- tools/GameSerializer.cpp

## 6.35 GoldComponent Struct Reference

Represents a gold amount an entity is holding, be it a gold seam, worker minion or gold depository.

```
#include <Components.hpp>
```

### Public Member Functions

- **GoldComponent** (tdt::uint max=0, tdt::uint curr=0)
- **GoldComponent** (const [GoldComponent](#) &)=default
- **GoldComponent** ([GoldComponent](#) &&)=default
- [GoldComponent](#) & **operator=** (const [GoldComponent](#) &)=default
- [GoldComponent](#) & **operator=** ([GoldComponent](#) &&)=default

### Public Attributes

- tdt::uint **max\_amount**
- tdt::uint **curr\_amount**

## Static Public Attributes

- static constexpr int **type** = 21

### 6.35.1 Detailed Description

Represents a gold amount an entity is holding, be it a gold seam, worker minion or gold depository.

Definition at line 528 of file Components.hpp.

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

- Components.hpp

## 6.36 GraphicsComponent Struct Reference

Holds info related to the Ogre3D rendering library.

```
#include <Components.hpp>
```

## Public Member Functions

- **GraphicsComponent** (std::string &&me="ogrehead.mesh", std::string &&ma="Ogre", bool v=true, bool manual=false, Ogre::Vector3 sc=Ogre::Vector3{0, 0, 0})
- **GraphicsComponent** (const [GraphicsComponent](#) &)=default
- **GraphicsComponent** ([GraphicsComponent](#) &&)=default
- [GraphicsComponent](#) & **operator=** (const [GraphicsComponent](#) &)=default
- [GraphicsComponent](#) & **operator=** ([GraphicsComponent](#) &&)=default

## Public Attributes

- std::string **mesh**
- std::string **material**
- bool **visible**
- Ogre::SceneNode \* **node**
- Ogre::Entity \* **entity**
- bool **manual\_scaling**
- Ogre::Vector3 **scale**

## Static Public Attributes

- static constexpr int **type** = 3



### 6.36.1 Detailed Description

Holds info related to the Ogre3D rendering library.

Definition at line 101 of file Components.hpp.

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

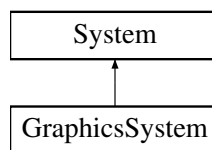
- Components.hpp

## 6.37 GraphicsSystem Class Reference

[System](#) that performs all graphics related updates.

```
#include <GraphicsSystem.hpp>
```

Inheritance diagram for GraphicsSystem:



### Public Member Functions

- [GraphicsSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~GraphicsSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Performs all graphics updates.*
- void [set\\_update\\_period](#) (tdt::real)  
*Sets the time period before the next update.*
- tdt::real [get\\_update\\_period](#) () const  
*Returns the time period between updates.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system that contains entities this system is working with.*
- tdt::real [update\\_timer\\_](#)  
*Used to avoid per frame updates and allows dynamic update periods.*
- tdt::real [update\\_period\\_](#)

### 6.37.1 Detailed Description

[System](#) that performs all graphics related updates.

Definition at line 10 of file GraphicsSystem.hpp.

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 GraphicsSystem::GraphicsSystem ( [EntitySystem](#) & *ents* )

Constructor.

#### Parameters

<i>The</i>	game's entity system.
------------	-----------------------

Definition at line 4 of file GraphicsSystem.cpp.

#### 6.37.2.2 GraphicsSystem::~GraphicsSystem ( ) [default]

Destructor.

### 6.37.3 Member Function Documentation

#### 6.37.3.1 tdt::real GraphicsSystem::get\_update\_period ( ) const

Returns the time period between updates.

Definition at line 45 of file GraphicsSystem.cpp.

#### 6.37.3.2 void GraphicsSystem::set\_update\_period ( tdt::real val )

Sets the time period before the next update.

#### Parameters

<i>The</i>	new period.
------------	-------------

Definition at line 40 of file GraphicsSystem.cpp.

#### 6.37.3.3 void GraphicsSystem::update ( tdt::real delta ) [override],[virtual]

Performs all graphics updates.

#### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 8 of file GraphicsSystem.cpp.

### 6.37.4 Member Data Documentation

#### 6.37.4.1 EntitySystem& GraphicsSystem::entities\_ [private]

Entity system that contains entities this system is working with.

Definition at line 45 of file GraphicsSystem.hpp.

## 6.37.4.2 tdt::real GraphicsSystem::update\_timer\_ [private]

Used to avoid per frame updates and allows dynamic update periods.

Definition at line 51 of file GraphicsSystem.hpp.

The documentation for this class was generated from the following files:

- systems/GraphicsSystem.hpp
- systems/GraphicsSystem.cpp

## 6.38 Grid Class Reference

Class representing the pathfinding grid.

```
#include <Grid.hpp>
```

### Public Member Functions

- bool [in\\_board](#) (tdt::uint) const  
*Returns true if a given node is in the grid.*
- const std::set< tdt::uint > & [get\\_freed](#) () const  
*Returns a constant reference to the list of freed nodes.*
- const std::set< tdt::uint > & [get\\_unfreed](#) () const  
*Returns a constant reference to the list of unfreed nodes.*
- void [clear\\_freed](#) ()  
*Removes all nodes from the list of freed nodes.*
- void [clear\\_unfreed](#) ()  
*Removes all nodes from the list of unfreed nodes.*
- tdt::uint [add\\_node](#) (EntitySystem &, Ogre::Vector2)  
*Created a new node at the given position.*
- void [add\\_freed](#) (tdt::uint)  
*Adds a given node to the list of the freed nodes.*
- void [add\\_unfreed](#) (tdt::uint)  
*Adds a given node to the list of the unfreed nodes.*
- void [remove\\_node](#) (tdt::uint)  
*Removes a given node from the node list.*
- tdt::uint [get\\_node](#) (tdt::uint, tdt::uint) const  
*Returns the ID of a node at a given position in the grid.*
- tdt::uint [get\\_node\\_from\\_position](#) (tdt::real, tdt::real) const  
*Returns the ID of a node that is closed to a given world coordinate.*
- void [create\\_graph](#) (EntitySystem &, Ogre::Vector2, tdt::uint, tdt::uint, tdt::real)  
*Generates a grid graph with the given parameters to be used for pathfinding.*
- tdt::real [get\\_distance](#) () const  
*Returns the distance between two nodes in the four non-diagonal directions.*
- tdt::uint [get\\_random\\_free\\_node](#) () const  
*Breif: Returns a random node within the graph.*
- Ogre::Vector2 [get\\_center\\_position](#) (EntitySystem &) const  
*Returns the 2D position of the central node of the grid (or one of them if the grid has even dimensions).*

- `Grid (const Grid &)=delete`  
*Since there should be only one grid at all times accesible from the `Grid::instance` method, all copy/move operations are disabled for this class.*
- `Grid & operator= (const Grid &)=delete`
- `Grid (Grid &&)=delete`
- `Grid & operator= (Grid &&)=delete`
- `bool place_at_random_free_node (EntitySystem &, tdt::uint)`  
*Places a given entity at a random node that is not obstructed by a building.*
- `bool distribute_to_adjacent_free_nodes (EntitySystem &, tdt::uint, const std::vector< tdt::uint > &)`  
*Distributes a given set of entities on free nodes adjacent to a given central node.*

## Static Public Member Functions

- `static Grid & instance ()`  
*Returns a reference to the static instance of this class.*

## Private Member Functions

- `Grid ()=default`  
*Constructor.*
- `~Grid ()`  
*Destructor.*
- `void link_ (tdt::uint, std::vector< GridNodeComponent * > &)`  
*Generates a neighbour list for a given node (thus linking it to the graph).*

## Private Attributes

- `std::vector< tdt::uint > nodes_`  
*Vector containing the IDs of the nodes in the grid, basically representing a 2D matrix stored in a 1D container.*
- `std::set< tdt::uint > freed_`  
*Auxiliary vectors containing IDs of the nodes that have been freed/unfreed on last frame.*
- `std::set< tdt::uint > unfreed_`
- `tdt::uint width_`  
*Dimensions of the grid in node count.*
- `tdt::uint height_`
- `tdt::real distance_`  
*Distance between two adjascent nodes.*
- `tdt::uint starting_index_`  
*ID of the first node, this allows for the node IDs to be outside the (0, width\_ \* height\_) range,.*
- `Ogre::Vector2 start_`  
*Coordinates of the starting node of the grid.*
- `std::vector< tdt::uint > free_nodes_`  
*Used for easier returning of a random free node.*

## Friends

- `class GameSerializer`  
*`GameSerializer` is a friend class so that it can easily access the grid realted data (like dimensions and node distance) when saving the game.*

### 6.38.1 Detailed Description

Class representing the pathfinding grid.

Definition at line 13 of file Grid.hpp.

### 6.38.2 Constructor & Destructor Documentation

#### 6.38.2.1 `Grid::Grid ( const Grid & ) [delete]`

Since there should be only one grid at all times accesible from the [Grid::instance](#) method, all copy/move operations are disabled for this class.

#### 6.38.2.2 `Grid::Grid ( ) [private],[default]`

Constructor.

Kept private since there should be only one grid at all times.

#### 6.38.2.3 `Grid::~Grid ( ) [inline],[private]`

Destructor.

Definition at line 162 of file Grid.hpp.

### 6.38.3 Member Function Documentation

#### 6.38.3.1 `void Grid::add_freed ( tdt::uint id )`

Adds a given node to the list of the freed nodes.

Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 50 of file Grid.cpp.

#### 6.38.3.2 `tdt::uint Grid::add_node ( EntitySystem & ents, Ogre::Vector2 pos )`

Created a new node at the given position.

Parameters

<a href="#">EntitySystem</a>	that contains the node.
<i>2D</i>	position of the node.

Definition at line 34 of file Grid.cpp.

#### 6.38.3.3 void Grid::add\_unfreed ( tdt::uint *id* )

Adds a given node to the list of the unfreed nodes.

##### Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 70 of file Grid.cpp.

#### 6.38.3.4 void Grid::clear\_freed ( )

Removes all nodes from the list of freed nodes.

Definition at line 24 of file Grid.cpp.

#### 6.38.3.5 void Grid::clear\_unfreed ( )

Removes all nodes from the list of unfreed nodes.

Definition at line 29 of file Grid.cpp.

#### 6.38.3.6 void Grid::create\_graph ( EntitySystem & *ents*, Ogre::Vector2 *start*, tdt::uint *w*, tdt::uint *h*, tdt::real *d* )

Generates a grid graph with the given parameters to be used for pathfinding.

##### Parameters

<i>EntitySystem</i>	that will contain the nodes.
<i>Starting</i>	position (x,z axes) of the grid.
<i>Width</i>	of the graph (in node count).
<i>Height</i>	of the graph (in node count).
<i>Distance</i>	between adjacent nodes.

Definition at line 112 of file Grid.cpp.

#### 6.38.3.7 bool Grid::distribute\_to\_adjacent\_free\_nodes ( EntitySystem & *ents*, tdt::uint *node*, const std::vector< tdt::uint > & *ids* )

Distributes a given set of entities on free nodes adjacent to a given central node.

Returns true if the placement was possible, false otherwise.

## Parameters

<i>Entity</i>	system containing the entitites.
<i>ID</i>	of the central node.
<i>Vector</i>	of IDs of the entities.

Definition at line 186 of file Grid.cpp.

#### 6.38.3.8 `Ogre::Vector2 Grid::get_center_position ( EntitySystem & ents ) const`

Returns the 2D position of the central node of the grid (or one of them if the grid has even dimensions).

## Parameters

<a href="#"><i>EntitySystem</i></a>	that contains components of the nodes.
-------------------------------------	--

Definition at line 171 of file Grid.cpp.

#### 6.38.3.9 `tdt::real Grid::get_distance ( ) const`

Returns the distance between two nodes in the four non-diagonal directions.

Definition at line 151 of file Grid.cpp.

#### 6.38.3.10 `const std::set< tdt::uint > & Grid::get_freed ( ) const`

Returns a constant reference to the list of freed nodes.

Definition at line 14 of file Grid.cpp.

#### 6.38.3.11 `tdt::uint Grid::get_node ( tdt::uint x, tdt::uint y ) const`

Returns the ID of a node at a given position in the grid.

## Parameters

<i>Column</i>	number.
<i>Row</i>	number.

Definition at line 86 of file Grid.cpp.

#### 6.38.3.12 `tdt::uint Grid::get_node_from_position ( tdt::real x, tdt::real y ) const`

Returns the ID of a node that is closed to a given world coordinate.

## Parameters

<i>X</i>	axis coordinate.
<i>Z</i>	axis coordinate.

## Note

Adding the ability to specify in what direction the node must be might be beneficial for pathfinding.

Definition at line 94 of file Grid.cpp.

#### 6.38.3.13 tdt::uint Grid::get\_random\_free\_node ( ) const

Brief: Returns a random node within the graph.

Definition at line 163 of file Grid.cpp.

#### 6.38.3.14 const std::set< tdt::uint > & Grid::get\_unfreed ( ) const

Returns a constant reference to the list of unfreed nodes.

Definition at line 19 of file Grid.cpp.

#### 6.38.3.15 bool Grid::in\_board ( tdt::uint *id* ) const

Returns true if a given node is in the grid.

## Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 9 of file Grid.cpp.

#### 6.38.3.16 Grid & Grid::instance ( ) [static]

Returns a reference to the static instance of this class.

## Note

Handles initialization and safe destruction by itself.

Definition at line 156 of file Grid.cpp.

#### 6.38.3.17 void Grid::link\_ ( tdt::uint *index*, std::vector< GridNodeComponent \* > & *comps* ) [private]

Generates a neighbour list for a given node (thus linking it to the graph).



## Parameters

<i>ID</i>	of the node.
<i>Auxiliary</i>	vector containing component pointers for fast access. (This method will ever be called only in the GridSystem::create_graph method, which already has such a vector and so it's used here too.)

Definition at line 218 of file Grid.cpp.

#### 6.38.3.18 bool Grid::place\_at\_random\_free\_node ( EntitySystem & ents, tdt::uint id )

Places a given entity at a random node that is not obstructed by a building.

Returns true if the placement was possible, false otherwise.

## Parameters

<i>Entity</i>	system containing the entity.
<i>ID</i>	of the entity.

Definition at line 177 of file Grid.cpp.

#### 6.38.3.19 void Grid::remove\_node ( tdt::uint id )

Removes a given node from the node list.

## Parameters

<i>ID</i>	of the node. TODO: Possibly implement "unlink"?
-----------	---

Definition at line 80 of file Grid.cpp.

### 6.38.4 Friends And Related Function Documentation

#### 6.38.4.1 friend class GameSerializer [friend]

[GameSerializer](#) is a friend class so that it can easily access the grid related data (like dimensions and node distance) when saving the game.

Definition at line 19 of file Grid.hpp.

### 6.38.5 Member Data Documentation

#### 6.38.5.1 tdt::real Grid::distance\_ [private]

Distance between two adjacent nodes.

Definition at line 195 of file Grid.hpp.

#### 6.38.5.2 `std::vector<tdt::uint> Grid::free_nodes_` [private]

Used for easier returning of a random free node.

Definition at line 213 of file Grid.hpp.

#### 6.38.5.3 `std::set<tdt::uint> Grid::freed_` [private]

Auxiliary vectors containing IDs of the nodes that have been freed/unfreed on last frame.

Used for pathfinding correction and structure model changes.

Definition at line 184 of file Grid.hpp.

#### 6.38.5.4 `std::vector<tdt::uint> Grid::nodes_` [private]

Vector containing the IDs of the nodes in the grid, basically representing a 2D matrix stored in a 1D container.

Definition at line 177 of file Grid.hpp.

#### 6.38.5.5 `Ogre::Vector2 Grid::start_` [private]

Coordinates of the starting node of the grid.

(This 2D vector contains X and Z coordinates despite it's second member being names Y).

Definition at line 208 of file Grid.hpp.

#### 6.38.5.6 `tdt::uint Grid::starting_index_` [private]

ID of the first node, this allows for the node IDs to be outside the (0, width\_ \* height\_) range,.

Definition at line 201 of file Grid.hpp.

#### 6.38.5.7 `tdt::uint Grid::width_` [private]

Dimensions of the grid in node count.

(Actual dimensions = dimensions \* distance.)

Definition at line 190 of file Grid.hpp.

The documentation for this class was generated from the following files:

- tools/Grid.hpp
- tools/Grid.cpp

## 6.39 GridNodeComponent Struct Reference

Holds GridNode's neighbour nodes.

```
#include <Components.hpp>
```

### Public Member Functions

- **GridNodeComponent** (std::array< tdt::uint, neighbour\_count > neigh=std::array< tdt::uint, neighbour\_count >{}, bool f=true, tdt::uint pos\_x=0, tdt::uint pos\_y=0, tdt::uint res=Component::NO\_ENTITY)
- **GridNodeComponent** (const [GridNodeComponent](#) &)=default
- **GridNodeComponent** ([GridNodeComponent](#) &&)=default
- [GridNodeComponent](#) & **operator=** (const [GridNodeComponent](#) &)=default
- [GridNodeComponent](#) & **operator=** ([GridNodeComponent](#) &&)=default

### Public Attributes

- std::array< tdt::uint, neighbour\_count > **neighbours**
- bool **free**
- tdt::uint **x**
- tdt::uint **y**
- tdt::uint **resident**

### Static Public Attributes

- static constexpr int **type** = 12
- static constexpr tdt::uint **neighbour\_count** = 9

#### 6.39.1 Detailed Description

Holds GridNode's neighbour nodes.

#### Note

The neighbours are set to the maximum value of tdt::uint to fix a state when one or more neighbours weren't set (won't have that many nodes so the A\* algorithm will ignore them).

Definition at line 321 of file Components.hpp.

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

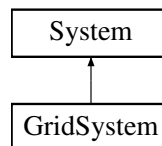
- Components.hpp

## 6.40 GridSystem Class Reference

Represents the pathfinding graph used by the game and provides several methods related to pathfinding that can be used in Lua.

```
#include <GridSystem.hpp>
```

Inheritance diagram for GridSystem:



### Public Member Functions

- [GridSystem](#) ([EntitySystem](#) &, [Ogre::SceneManager](#) &)  
*Constructor.*
- [~GridSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Checks if any nodes were freed or unfreed and if so, corrects any path that had those nodes in it.*
- void [create\\_graphics](#) ()  
*Creates and initializes Ogre models for nodes, which allows the developer to display them for testing purposes.*
- void [delete\\_graphics](#) ()  
*Deletes Ogre models of all nodes.*
- void [set\\_visible](#) (bool)  
*Changes the visibility status of the nodes if the graphics have been created already, does nothing otherwise.*
- bool [is\\_visible](#) () const  
*Returns true if the the grid models are visible, false otherwise.*
- void [place\\_structure](#) (tdt::uint, tdt::uint, tdt::uint)  
*Places a structure (building, wall...) by changing the nodes it is placed on to not free, managing residences etc.*

### Private Member Functions

- void [update\\_neighbours](#) (tdt::uint)  
*Updates the nodes resident's alignment (if possible) when it's neighbour was freed/unfreed.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- [Ogre::SceneManager](#) & [scene\\_mgr\\_](#)  
*Reference to the game's main scene manager (used for graphics creation).*
- bool [graphics\\_loaded\\_](#)  
*Determine if the graphics have been loaded and if the graph is visible (which is only relevant if the former is true).*
- bool [graph\\_visible\\_](#)

### 6.40.1 Detailed Description

Represents the pathfinding graph used by the game and provides several methods related to pathfinding that can be used in Lua.

Definition at line 11 of file GridSystem.hpp.

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 GridSystem::GridSystem ( EntitySystem & *ents*, Ogre::SceneManager & *scene* )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
<i>Reference</i>	to the game's main scene manager.

Definition at line 12 of file GridSystem.cpp.

#### 6.40.2.2 GridSystem::~~GridSystem ( ) [default]

Destructor.

### 6.40.3 Member Function Documentation

#### 6.40.3.1 void GridSystem::create\_graphics ( )

Creates and initializes Ogre models for nodes, which allows the developer to display them for testing purposes.

Definition at line 80 of file GridSystem.cpp.

#### 6.40.3.2 void GridSystem::delete\_graphics ( )

Deletes Ogre models of all nodes.

Definition at line 106 of file GridSystem.cpp.

#### 6.40.3.3 bool GridSystem::is\_visible ( ) const

Returns true if the the grid models are visible, false otherwise.

Definition at line 130 of file GridSystem.cpp.

#### 6.40.3.4 void GridSystem::place\_structure ( tdt::uint *ent\_id*, tdt::uint *node\_id*, tdt::uint *radius* )

Places a structure (building, wall...) by changing the nodes it is placed on to not free, managing residences etc.

## Parameters

<i>ID</i>	of the structure.
<i>ID</i>	of the central node.
<i>Radius</i>	of the structure.

Definition at line 138 of file GridSystem.cpp.

#### 6.40.3.5 void GridSystem::set\_visible ( bool *on\_off* )

Changes the visibility status of the nodes if the graphics have been created already, does nothing otherwise.

## Parameters

<i>The</i>	new visibility status.
------------	------------------------

Definition at line 114 of file GridSystem.cpp.

#### 6.40.3.6 void GridSystem::update ( tdt::real ) [override], [virtual]

Checks if any nodes were freed or unfreed and if so, corrects any path that had those nodes in it.

## Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 17 of file GridSystem.cpp.

#### 6.40.3.7 void GridSystem::update\_neighbours\_ ( tdt::uint *id* ) [private]

Updates the nodes resident's alignment (if possible) when it's neighbour was freed/unfreed.

## Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 168 of file GridSystem.cpp.

### 6.40.4 Member Data Documentation

#### 6.40.4.1 EntitySystem& GridSystem::entities\_ [private]

Reference to the game's entity system.

Definition at line 76 of file GridSystem.hpp.

#### 6.40.4.2 bool GridSystem::graphics\_loaded\_ [private]

Determine if the graphics have been loaded and if the graph is visible (which is only relevant if the former is true).

Definition at line 88 of file GridSystem.hpp.

#### 6.40.4.3 Ogre::SceneManager& GridSystem::scene\_mgr\_ [private]

Reference to the game's main scene manager (used for graphics creation).

Definition at line 81 of file GridSystem.hpp.

The documentation for this class was generated from the following files:

- systems/GridSystem.hpp
- systems/GridSystem.cpp

## 6.41 GUI Class Reference

Represents the game's main graphical user interface (i.e.

```
#include <GUI.hpp>
```

### Public Member Functions

- [~GUI](#) ()  
*Destructor.*
- void [init](#) (Game \*)  
*Initializes the selection by loading the layout and registering all event handlers.*
- void [set\\_visible](#) (bool)  
*Set's the GUI's visibility status.*
- bool [is\\_visible](#) () const  
*Returns true if the entire GUI is visible, false otherwise.*
- void [set\\_visible](#) (const std::string &, bool)  
*Set's the visibility status of a particular window.*
- bool [is\\_visible](#) (const std::string &) const  
*Returns the visibility status of a particular window.*
- void [show\\_load\\_save\\_dialog](#) (const std::string &)  
*Shows the load/save dialog window.*
- CEGUI::Window \* [get\\_window](#) ()  
*Returns a pointer to the root window.*
- CEGUI::Window \* [get\\_window](#) (const std::string &)  
*Returns a pointer to a given subwindow of the root window.*
- Console & [get\\_console](#) ()  
*Returns a reference to the game's dev console.*
- EntityTracker & [get\\_tracker](#) ()  
*Returns a reference to the entity tracker.*
- GameLog & [get\\_log](#) ()

- Returns a reference to the game's log.*
- [BuilderWindow](#) & [get\\_builder](#) ()
  - Returns a reference to the builder window.*
- [TopBar](#) & [get\\_top\\_bar](#) ()
  - Returns a reference to the top bar.*
- [ResearchWindow](#) & [get\\_research](#) ()
  - Returns a reference to the research window.*
- [SpellCastingWindow](#) & [get\\_spell\\_casting](#) ()
  - Returns a reference to the spell casting window.*
- [MessageToPlayerWindow](#) & [get\\_message](#) ()
  - Returns a reference to the message to player window.*
- [OptionsWindow](#) & [get\\_options](#) ()
  - Returns a reference to the options window.*
- bool [escape\\_pressed](#) ()
  - Notifies the [GUI](#) that the escape key was pressed so that it can close windows if needed.*
- void [set\\_curr\\_tool\\_visible](#) (bool)
  - Sets the visibility status of the current tool window.*
- void [set\\_curr\\_tool](#) (const std::string &)
  - Sets the current tool window.*
- const std::string & [get\\_curr\\_tool](#) ()
  - Returns the name of the current tool window.*
- [GUI](#) (const [GUI](#) &)=delete
- [GUI](#) ([GUI](#) &&)=delete
- [GUI](#) & **operator=** (const [GUI](#) &)=delete
- [GUI](#) & **operator=** ([GUI](#) &&)=delete

## Static Public Member Functions

- static [GUI](#) & [instance](#) ()
  - Returns the singleton instance.*

## Private Member Functions

- [GUI](#) ()
  - Constructor, private because of the singleton pattern.*
- void [list\\_directory](#) (const std::string &, CEGUI::Listbox &, bool=false)
  - Fills a given list box with the names of all files in a given directory.*

## Private Attributes

- CEGUI::Window \* [window\\_](#)
  - Pointer to the root window of the layout.*
- std::string [curr\\_tool\\_](#)
  - Name of the current tool in the tools subwindow.*
- [Game](#) \* [game\\_](#)
  - Pointer to the game instance used by button event handlers.*
- [EntityTracker](#) [tracker\\_](#)
  - ID of the entity that is currently tracked by the entity viewer.*
- [Console](#) [console\\_](#)



- [Game's developer console.](#)
- [GameLog log\\_](#)  
*Game's log, used to show messages to the player.*
- [BuilderWindow builder\\_](#)  
*Allows the player to place buildings.*
- [TopBar top\\_bar\\_](#)  
*Shows game info at the top of the screen.*
- [ResearchWindow research\\_](#)  
*Game's research window that allows the player to buy buildings, units and spells.*
- [SpellCastingWindow spell\\_casting\\_](#)  
*Allows the player to cast spells.*
- [CEGUI::Window \\* menu\\_](#)  
*Allows easier access to the menu subwindow.*
- [MessageToPlayerWindow message\\_](#)  
*Allows the game to show a message to the player with any of the following buttons: OK, YES, NO.*
- [OptionsWindow options\\_](#)  
*Allows to change the resolution, window mode and key bindings.*

## Friends

- class **OptionsWindow**
- void **action::QUICK\_LOAD** ()
- void **action::QUICK\_SAVE** ()
- void **action::RESET\_CAMERA** ()

### 6.41.1 Detailed Description

Represents the game's main graphical user interface (i.e.

any windows except for the development ones like the console or entity creator).

Definition at line 26 of file GUI.hpp.

### 6.41.2 Constructor & Destructor Documentation

#### 6.41.2.1 GUI::~GUI( ) [inline]

Destructor.

Definition at line 36 of file GUI.hpp.

#### 6.41.2.2 GUI::GUI( const GUI & ) [delete]

#### Note

Since VS2015 seems to have some problems with C++ standard (generates default copy/move constructors and operators even if default constructor is created), these constructors/operators are explicitly deleted.

#### 6.41.2.3 GUI::GUI( ) [private]

Constructor, private because of the singleton pattern.

Definition at line 7 of file GUI.cpp.

### 6.41.3 Member Function Documentation

#### 6.41.3.1 bool GUI::escape\_pressed( )

Notifies the [GUI](#) that the escape key was pressed so that it can close windows if needed.

Returns true if anything has been closed, false otherwise.

##### Note

CEGUI event system does not seem to work properly :/

This will allow the return from the main menu (which does not happend when a submenu window is visible).

Definition at line 414 of file GUI.cpp.

#### 6.41.3.2 BuilderWindow & GUI::get\_builder( )

Returns a reference to the builder window.

Definition at line 384 of file GUI.cpp.

#### 6.41.3.3 Console & GUI::get\_console( )

Returns a reference to the game's dev console.

Definition at line 369 of file GUI.cpp.

#### 6.41.3.4 const std::string & GUI::get\_curr\_tool( )

Returns the name of the current tool window.

Definition at line 465 of file GUI.cpp.

#### 6.41.3.5 GameLog & GUI::get\_log( )

Returns a reference to the game's log.

Definition at line 379 of file GUI.cpp.

**6.41.3.6 MessageToPlayerWindow & GUI::get\_message ( )**

Returns a reference to the message to player window.

Definition at line 404 of file GUI.cpp.

**6.41.3.7 OptionsWindow & GUI::get\_options ( )**

Returns a reference to the options window.

Definition at line 409 of file GUI.cpp.

**6.41.3.8 ResearchWindow & GUI::get\_research ( )**

Returns a reference to the research window.

Definition at line 394 of file GUI.cpp.

**6.41.3.9 SpellCastingWindow & GUI::get\_spell\_casting ( )**

Returns a reference to the spell casting window.

Definition at line 399 of file GUI.cpp.

**6.41.3.10 TopBar & GUI::get\_top\_bar ( )**

Returns a reference to the top bar.

Definition at line 389 of file GUI.cpp.

**6.41.3.11 EntityTracker & GUI::get\_tracker ( )**

Returns a reference to the entity tracker.

Definition at line 374 of file GUI.cpp.

**6.41.3.12 CEGUI::Window \* GUI::get\_window ( )**

Returns a pointer to the root window.

Definition at line 359 of file GUI.cpp.

**6.41.3.13 CEGUI::Window \* GUI::get\_window ( const std::string & name )**

Returns a pointer to a given subwindow of the root window.

## Parameters

<i>Name</i>	of the window.
-------------	----------------

Definition at line 364 of file GUI.cpp.

#### 6.41.3.14 void GUI::init ( **Game** \* *game* )

Initializes the selection by loading the layout and registering all event handlers.

## Parameters

<i>Pointer</i>	to the game object, which is used by the event handlers (like the quit button etc).
----------------	---

The options menu was stored in it's own separate file for easier design.

MAIN MENU

TOOL SELECTION

MENU

Definition at line 14 of file GUI.cpp.

#### 6.41.3.15 GUI & GUI::instance ( ) [static]

Returns the singleton instance.

Definition at line 335 of file GUI.cpp.

#### 6.41.3.16 bool GUI::is\_visible ( ) const

Returns true if the entire [GUI](#) is visible, false otherwise.

Definition at line 320 of file GUI.cpp.

#### 6.41.3.17 bool GUI::is\_visible ( const std::string & *wname* ) const

Returns the visibility status of a particular window.

## Parameters

<i>Name</i>	(path) of the window, without the root window prefix.
-------------	---

Definition at line 330 of file GUI.cpp.

6.41.3.18 void GUI::list\_directory ( const std::string & *dir*, CEGUI::Listbox & *box*, bool *strip\_ext* = false ) [private]

Fills a given list box with the names of all files in a given directory.

Parameters

<i>Name</i>	of the directory.
<i>List</i>	box to be filled.
<i>If</i>	true, the .lua extension will be cut from the file name if present.

Definition at line 470 of file GUI.cpp.

6.41.3.19 void GUI::set\_curr\_tool ( const std::string & *val* )

Sets the current tool window.

Parameters

<i>The</i>	name of the tool. (MENU, SPELL, BUILD)
------------	--

Definition at line 459 of file GUI.cpp.

6.41.3.20 void GUI::set\_curr\_tool\_visible ( bool *val* )

Sets the visibility status of the current tool window.

Parameters

<i>The</i>	new visibility status.
------------	------------------------

Definition at line 454 of file GUI.cpp.

6.41.3.21 void GUI::set\_visible ( bool *val* )

Set's the GUI's visibility status.

Parameters

<i>The</i>	new visibility status. TODO: Toggle GUI with a hotkey for screen capturing?
------------	---

Definition at line 315 of file GUI.cpp.

6.41.3.22 void GUI::set\_visible ( const std::string & *wname*, bool *val* )

Set's the visibility status of a particular window.

## Parameters

<i>Name</i>	(path) of the window, without the root window prefix. (e.g. "TOOLS/TOOL_SELECTION/SPELL_SELECTION")
<i>The</i>	new visibility status.

Definition at line 325 of file GUI.cpp.

#### 6.41.3.23 void GUI::show\_load\_save\_dialog ( const std::string & type )

Shows the load/save dialog window.

## Parameters

<i>Either</i>	"SAVE" or "LOAD", will determine the functionality of the window.
---------------	---

Definition at line 341 of file GUI.cpp.

### 6.41.4 Member Data Documentation

#### 6.41.4.1 BuilderWindow GUI::builder\_ [private]

Allows the player to place buildings.

Definition at line 226 of file GUI.hpp.

#### 6.41.4.2 Console GUI::console\_ [private]

[Game](#)'s developer console.

Definition at line 216 of file GUI.hpp.

#### 6.41.4.3 std::string GUI::curr\_tool\_ [private]

Name of the current tool in the tools subwindow.

Definition at line 201 of file GUI.hpp.

#### 6.41.4.4 Game\* GUI::game\_ [private]

Pointer to the game instance used by button event handlers.

Definition at line 206 of file GUI.hpp.

**6.41.4.5   GameLog** `GUI::log_` [private]

`Game`'s log, used to show messages to the player.

Definition at line 221 of file GUI.hpp.

**6.41.4.6   CEGUI::Window\*** `GUI::menu_` [private]

Allows easier access to the menu subwindow.

Definition at line 247 of file GUI.hpp.

**6.41.4.7   MessageToPlayerWindow** `GUI::message_` [private]

Allows the game to show a message to the player with any of the following buttons: OK, YES, NO.

Definition at line 253 of file GUI.hpp.

**6.41.4.8   OptionsWindow** `GUI::options_` [private]

Allows to change the resolution, window mode and key bindings.

Definition at line 259 of file GUI.hpp.

**6.41.4.9   ResearchWindow** `GUI::research_` [private]

`Game`'s research window that allows the player to buy buildings, units and spells.

Definition at line 237 of file GUI.hpp.

**6.41.4.10   SpellCastingWindow** `GUI::spell_casting_` [private]

Allows the player to cast spells.

Definition at line 242 of file GUI.hpp.

**6.41.4.11   TopBar** `GUI::top_bar_` [private]

Shows game info at the top of the screen.

Definition at line 231 of file GUI.hpp.

**6.41.4.12   EntityTracker** `GUI::tracker_` [private]

ID of the entity that is currently tracked by the entity viewer.

Definition at line 211 of file GUI.hpp.

#### 6.41.4.13 CEGUI::Window\* GUI::window\_ [private]

Pointer to the root window of the layout.

Definition at line 196 of file GUI.hpp.

The documentation for this class was generated from the following files:

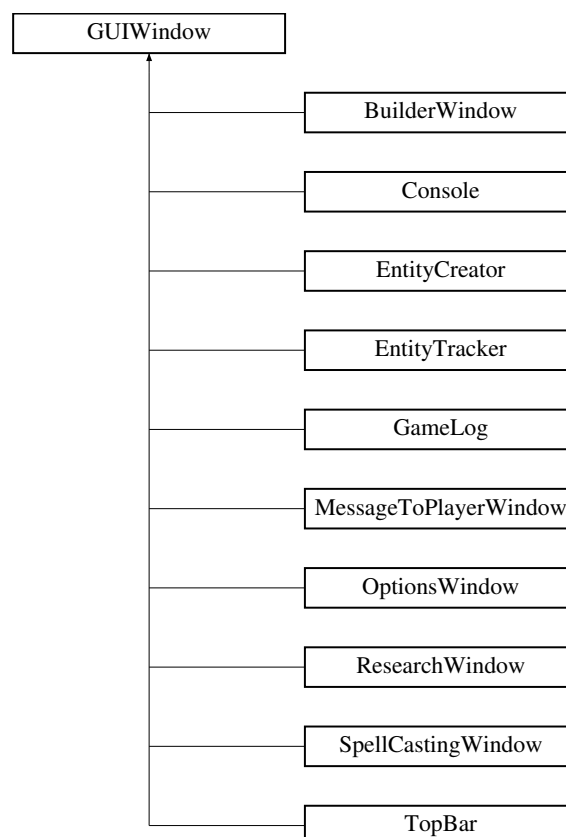
- gui/GUI.hpp
- gui/GUI.cpp

## 6.42 GUIWindow Class Reference

Abstract class that custom [GUI](#) windows inherit from, prevents unnecessary rewriting of common functions (like visibility setting and window\_ assignment on init).

```
#include <GUIWindow.hpp>
```

Inheritance diagram for GUIWindow:





## Public Member Functions

- [GUIWindow](#) ()  
*Constructor.*
- virtual [~GUIWindow](#) ()=default  
*Destructor.*
- void [init](#) (CEGUI::Window \*)  
*Initializes the window\_ variable and calls the protected init\_ function.*
- virtual void [set\\_visible](#) (bool)  
*Sets the visibility status of this window.*
- bool [is\\_visible](#) () const  
*Returns true if the window is visible, false otherwise.*

## Protected Member Functions

- virtual void [init\\_](#) ()=0  
*Specific init function for each inheriting class.*

## Protected Attributes

- CEGUI::Window \* [window\\_](#)  
*Root window.*

### 6.42.1 Detailed Description

Abstract class that custom [GUI](#) windows inherit from, prevents unnecessary rewriting of common functions (like visibility setting and window\_ assignment on init).

Definition at line 12 of file GUIWindow.hpp.

### 6.42.2 Constructor & Destructor Documentation

#### 6.42.2.1 GUIWindow::GUIWindow ( )

Constructor.

Definition at line 4 of file GUIWindow.cpp.

#### 6.42.2.2 virtual GUIWindow::~~GUIWindow ( ) [virtual],[default]

Destructor.

### 6.42.3 Member Function Documentation

#### 6.42.3.1 void GUIWindow::init ( CEGUI::Window \* w )

Initializes the window\_ variable and calls the protected init\_ function.

Definition at line 8 of file GUIWindow.cpp.

#### 6.42.3.2 virtual void GUIWindow::init\_ ( ) [protected], [pure virtual]

Specific init function for each inheriting class.

Implemented in [SpellCastingWindow](#), [Console](#), [ResearchWindow](#), [BuilderWindow](#), [EntityTracker](#), [EntityCreator](#), [MessageToPlayerWindow](#), [OptionsWindow](#), [GameLog](#), and [TopBar](#).

#### 6.42.3.3 bool GUIWindow::is\_visible ( ) const

Returns true if the window is visible, false otherwise.

Definition at line 19 of file GUIWindow.cpp.

#### 6.42.3.4 void GUIWindow::set\_visible ( bool val ) [virtual]

Sets the visibolity status of this window.

##### Parameters

<i>The</i>	new visibility status.
------------	------------------------

Reimplemented in [Console](#).

Definition at line 14 of file GUIWindow.cpp.

### 6.42.4 Member Data Documentation

#### 6.42.4.1 CEGUI::Window\* GUIWindow::window\_ [protected]

Root window.

Definition at line 47 of file GUIWindow.hpp.

The documentation for this class was generated from the following files:

- gui/GUIWindow.hpp
- gui/GUIWindow.cpp

## 6.43 util::HAS\_GOLD Struct Reference

Tests if a given entity has a gold component.

```
#include <Util.hpp>
```

### Public Member Functions

- [HAS\\_GOLD](#) ([EntitySystem](#) &)  
*Constructor.*
- [~HAS\\_GOLD](#) ()=default  
*Destructor.*
- bool [operator\(\)](#) (tdt::uint)  
*Tests if a given entity has a gold component.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing all tested entities.*

#### 6.43.1 Detailed Description

Tests if a given entity has a gold component.

Definition at line 132 of file Util.hpp.

#### 6.43.2 Constructor & Destructor Documentation

##### 6.43.2.1 util::HAS\_GOLD::HAS\_GOLD ( [EntitySystem](#) & *ents* )

Constructor.

Parameters

<i>Entity</i>	system containing all tested entities.
---------------	--

Definition at line 45 of file Util.cpp.

##### 6.43.2.2 util::HAS\_GOLD::~~HAS\_GOLD ( ) [default]

Destructor.

### 6.43.3 Member Function Documentation

#### 6.43.3.1 `bool util::HAS_GOLD::operator() ( tdt::uint id )`

Tests if a given entity has a gold component.

##### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 49 of file Util.cpp.

### 6.43.4 Member Data Documentation

#### 6.43.4.1 `EntitySystem& util::HAS_GOLD::entities_ [private]`

Entity system containing all tested entities.

Definition at line 155 of file Util.hpp.

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

- tools/Util.hpp
- tools/Util.cpp

## 6.44 `util::effect::HEAL_EFFECT` Struct Reference

Fully heals the entity it's called on.

```
#include <Effects.hpp>
```

### Public Member Functions

- [HEAL\\_EFFECT](#) ([EntitySystem](#) &)  
*Constructor.*
- [~HEAL\\_EFFECT](#) ()=default  
*Destructor.*
- void [operator\(\)](#) (tdt::uint)  
*Fully heals a given entity.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing the entities this effect will be called on.*

### 6.44.1 Detailed Description

Fully heals the entity it's called on.

Definition at line 61 of file Effects.hpp.

### 6.44.2 Constructor & Destructor Documentation

#### 6.44.2.1 util::effect::HEAL\_EFFECT::HEAL\_EFFECT ( EntitySystem & ents )

Constructor.

Parameters

<i>Entity</i>	system containing the entities this effect will be called on.
---------------	---

Definition at line 16 of file Effects.cpp.

#### 6.44.2.2 util::effect::HEAL\_EFFECT::~~HEAL\_EFFECT ( ) [default]

Destructor.

### 6.44.3 Member Function Documentation

#### 6.44.3.1 void util::effect::HEAL\_EFFECT::operator() ( tdt::uint id )

Fully heals a given entity.

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 20 of file Effects.cpp.

### 6.44.4 Member Data Documentation

#### 6.44.4.1 EntitySystem& util::effect::HEAL\_EFFECT::entities\_ [private]

Entity system containing the entities this effect will be called on.

Definition at line 86 of file Effects.hpp.

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

- tools/Effects.hpp
- tools/Effects.cpp

## 6.45 HealthComponent Struct Reference

Holds info about an entity's health and regeneration.

```
#include <Components.hpp>
```

### Public Member Functions

- **HealthComponent** (tdt::uint max=0, tdt::uint reg=0, tdt::uint def=0, bool a=true)
- **HealthComponent** (const [HealthComponent](#) &)=default
- **HealthComponent** ([HealthComponent](#) &&)=default
- [HealthComponent](#) & **operator=** (const [HealthComponent](#) &)=default
- [HealthComponent](#) & **operator=** ([HealthComponent](#) &&)=default

### Public Attributes

- tdt::uint **curr\_hp**
- tdt::uint **max\_hp**
- tdt::uint **regen**
- tdt::uint **defense**
- bool **alive**

### Static Public Attributes

- static constexpr int **type** = 1

### 6.45.1 Detailed Description

Holds info about an entity's health and regeneration.

Definition at line 55 of file Components.hpp.

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

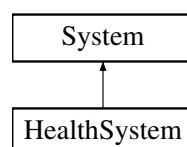
- Components.hpp

## 6.46 HealthSystem Class Reference

[System](#) that manages the regeneration and health of entities on each frame.

```
#include <HealthSystem.hpp>
```

Inheritance diagram for HealthSystem:



## Public Member Functions

- [HealthSystem \(EntitySystem &\)](#)  
*Constructor.*
- [~HealthSystem \(\)](#)=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Updates a the system by checking every valid entity's health (or death) status and applying health regeneration if necessary.*
- void [update\\_regen](#) (tdt::real)  
*Adds one to the regeneration timer, simulating continuous regeration, should be called once per frame before the update method.*
- void [set\\_regen\\_period](#) (tdt::real)  
*Sets the amount of time it takes for one regeneration tick to happen (in seconds).*
- tdt::real [get\\_regen\\_period](#) () const  
*Returns the amount of time it takes for one regeneration tick to happen (in seconds).*

## Private Attributes

- [EntitySystem & entities\\_](#)  
*Reference to the game's entity system.*
- tdt::real [regen\\_timer\\_](#)  
*Amount of frames since the last regeneration tick.*
- tdt::real [regen\\_period\\_](#)  
*Amount of frames per regeneration period.*
- bool [regen\\_](#)  
*True if this frame's update should renerate health.*

### 6.46.1 Detailed Description

[System](#) that manages the regeneration and health of entities on each frame.

Definition at line 11 of file HealthSystem.hpp.

### 6.46.2 Constructor & Destructor Documentation

#### 6.46.2.1 HealthSystem::HealthSystem ( EntitySystem & ent )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 6 of file HealthSystem.cpp.

#### 6.46.2.2 HealthSystem::~HealthSystem ( ) [default]

Destructor.

### 6.46.3 Member Function Documentation

#### 6.46.3.1 tdt::real HealthSystem::get\_regen\_period ( ) const

Returns the amount of time it takes for one regeneration tick to happen (in seconds).

Definition at line 43 of file HealthSystem.cpp.

#### 6.46.3.2 void HealthSystem::set\_regen\_period ( tdt::real val )

Sets the amount of time it takes for one regeneration tick to happen (in seconds).

##### Parameters

<i>The</i>	new regen period.
------------	-------------------

Definition at line 38 of file HealthSystem.cpp.

#### 6.46.3.3 void HealthSystem::update ( tdt::real delta ) [override],[virtual]

Updates a the system by checking every valid entity's health (or death) status and applying health regeneration if necessary.

##### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 11 of file HealthSystem.cpp.

#### 6.46.3.4 void HealthSystem::update\_regen ( tdt::real delta )

Adds one to the regeneration timer, simulating continuous regeration, should be called once per frame before the update method.

##### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Definition at line 23 of file HealthSystem.cpp.



## 6.46.4 Member Data Documentation

### 6.46.4.1 EntitySystem& HealthSystem::entities\_ [private]

Reference to the game's entity system.

Definition at line 56 of file HealthSystem.hpp.

### 6.46.4.2 bool HealthSystem::regen\_ [private]

True if this frame's update should regenerate health.

Definition at line 71 of file HealthSystem.hpp.

### 6.46.4.3 tdt::real HealthSystem::regen\_period\_ [private]

Amount of frames per regeneration period.

Definition at line 66 of file HealthSystem.hpp.

### 6.46.4.4 tdt::real HealthSystem::regen\_timer\_ [private]

Amount of frames since the last regeneration tick.

Definition at line 61 of file HealthSystem.hpp.

The documentation for this class was generated from the following files:

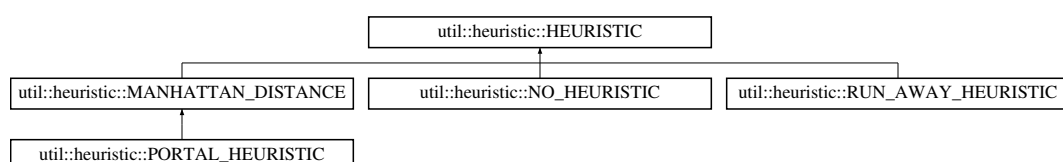
- systems/HealthSystem.hpp
- systems/HealthSystem.cpp

## 6.47 util::heuristic::HEURISTIC Struct Reference

Abstract parent of all heuristics.

```
#include <PathfindingAlgorithms.hpp>
```

Inheritance diagram for util::heuristic::HEURISTIC:



## Public Member Functions

- **HEURISTIC** ([EntitySystem](#) &ents)
- virtual tdt::real **get\_cost** (tdt::uint id1, tdt::uint id2)=0

## Protected Attributes

- [EntitySystem](#) & **entities\_**

### 6.47.1 Detailed Description

Abstract parent of all heuristics.

Inheritance hierarchy used instead of static functions (like in the case of `PATH_TYPES`) to allow for a heuristic to have a state.

Definition at line 188 of file `PathfindingAlgorithms.hpp`.

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

- `tools/PathfindingAlgorithms.hpp`

## 6.48 HomingComponent Struct Reference

Used for projectiles that are supposed to follow a target and deal damage when they hit it.

```
#include <Components.hpp>
```

## Public Member Functions

- **HomingComponent** (tdt::uint s=Component::NO\_ENTITY, tdt::uint t=Component::NO\_ENTITY, tdt::uint d=0)
- **HomingComponent** (const [HomingComponent](#) &)=default
- **HomingComponent** ([HomingComponent](#) &&)=default
- [HomingComponent](#) & **operator=** (const [HomingComponent](#) &)=default
- [HomingComponent](#) & **operator=** ([HomingComponent](#) &&)=default

## Public Attributes

- tdt::uint **source**
- tdt::uint **target**
- tdt::uint **dmg**

## Static Public Attributes

- static constexpr int **type** = 18

### 6.48.1 Detailed Description

Used for projectiles that are supposed to follow a target and deal damage when they hit it.

Definition at line 462 of file Components.hpp.

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

- Components.hpp

## 6.49 InputComponent Struct Reference

Holds info related to direct player input applied to an entity.

```
#include <Components.hpp>
```

### Public Member Functions

- **InputComponent** (std::string &&handler="ERROR.input\_handler")
- **InputComponent** (const [InputComponent](#) &)=default
- **InputComponent** ([InputComponent](#) &&)=default
- [InputComponent](#) & **operator=** (const [InputComponent](#) &)=default
- [InputComponent](#) & **operator=** ([InputComponent](#) &&)=default

### Public Attributes

- std::string **input\_handler**

### Static Public Attributes

- static constexpr int **type** = 7

### 6.49.1 Detailed Description

Holds info related to direct player input applied to an entity.

Definition at line 207 of file Components.hpp.

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

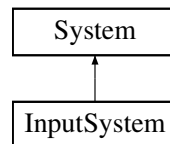
- Components.hpp

## 6.50 InputSystem Class Reference

[System](#) handling entities controlled by the player's keyboard input and changing the game's view mode (between 1st and 3rd person).

```
#include <InputSystem.hpp>
```

Inheritance diagram for InputSystem:



### Public Member Functions

- [InputSystem](#) ([EntitySystem](#) &, [OIS::Keyboard](#) &, [Ogre::Camera](#) &)  
*Constructor.*
- [~InputSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Handles the input for the entity that is currently in the first person mode.*
- bool [is\\_first\\_person](#) () const  
*Returns true if the game is in the first person mode, returns false otherwise.*
- void [set\\_first\\_person](#) (bool, tdt::uint=0)  
*Changes the first person mode status for an entity and also loads an [InputComponent](#) for it if it does not have it but has an [AIComponent](#) holding information about it's input\_handler method.*
- void [rebind](#) (int, int)  
*Rebinds a given key with an OIS key number.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- bool [first\\_person\\_](#)  
*Determines if the first person view is turned on.*
- tdt::uint [first\\_person\\_id\\_](#)  
*If the first person view is turned on, this holds ID of the entity being controlled.*
- [OIS::Keyboard](#) & [keyboard\\_](#)  
*Reference to the keyboard being used.*
- int [KEY\\_UP](#)  
*Current keybindings, allow rebinding.*
- int [KEY\\_DOWN](#)
- int [KEY\\_LEFT](#)
- int [KEY\\_RIGHT](#)
- [Ogre::Camera](#) & [cam\\_](#)  
*Reference to the game's main camera.*
- [Ogre::Vector3](#) [cam\\_position\\_](#)  
*Backup of the camera info for easy restoring once the game leaves the first person view.*

- `Ogre::Quaternion cam_orientation_`
- `std::unique_ptr< AIComponent > ai_backup_`  
*Backup of the AI component when entering first person view.*
- `std::unique_ptr< TaskHandlerComponent > task_backup_`  
*Backup of the task component when entering first person view.*
- `bool delete_input_`  
*Determines if the [InputComponent](#) of the entity being controlled in the first person view should be deleted once the game changes back to third person view.*

### 6.50.1 Detailed Description

[System](#) handling entities controlled by the player's keyboard input and changing the game's view mode (between 1st and 3rd person).

Definition at line 16 of file `InputSystem.hpp`.

### 6.50.2 Constructor & Destructor Documentation

#### 6.50.2.1 `InputSystem::InputSystem ( EntitySystem & ents, OIS::Keyboard & key, Ogre::Camera & cam )`

Constructor.

Parameters

<i>Reference</i>	to the game's <a href="#">EntitySystem</a> instance.
<i>Reference</i>	to the keyboard being used.
<i>Reference</i>	to the camera for it's manipulation during the 1st person mode.

Definition at line 5 of file `InputSystem.cpp`.

#### 6.50.2.2 `InputSystem::~~InputSystem ( ) [default]`

Destructor.

### 6.50.3 Member Function Documentation

#### 6.50.3.1 `bool InputSystem::is_first_person ( ) const`

Returns true if the game is in the first person mode, returns false otherwise.

Definition at line 56 of file `InputSystem.cpp`.

#### 6.50.3.2 `void InputSystem::rebind ( int key, int new_key )`

Rebinds a given key with an OIS key number.

Use `OIS::KC_W`, `OIS::KC_S`, `OIS::KC_A` and `OIS::KC_D` to determine which key should be rebounded.

## Parameters

<i>Key</i>	to be rebinded.
<i>The</i>	new key.

Definition at line 160 of file InputSystem.cpp.

#### 6.50.3.3 void InputSystem::set\_first\_person ( bool *on\_off*, tdt::uint *id* = 0 )

Changes the first person mode status for an entity and also loads an [InputComponent](#) for it if it does not have it but has an [AIComponent](#) holding information about it's input\_handler method.

(Backups the [AIComponent](#) in such cases and restores it when the entity leaves first person mode.)

Definition at line 61 of file InputSystem.cpp.

#### 6.50.3.4 void InputSystem::update ( tdt::real *delta* ) [override],[virtual]

Handles the input for the entity that is currently in the first person mode.

## Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 12 of file InputSystem.cpp.

### 6.50.4 Member Data Documentation

#### 6.50.4.1 std::unique\_ptr<AIComponent> InputSystem::ai\_backup\_ [private]

Backup of the AI component when entering first person view.

Definition at line 101 of file InputSystem.hpp.

#### 6.50.4.2 Ogre::Camera& InputSystem::cam\_ [private]

Reference to the game's main camera.

Definition at line 89 of file InputSystem.hpp.

#### 6.50.4.3 Ogre::Vector3 InputSystem::cam\_position\_ [private]

Backup of the camera info for easy restoring once the game leaves the first person view.

Definition at line 95 of file InputSystem.hpp.

#### 6.50.4.4 `bool InputSystem::delete_input_ [private]`

Determines if the [InputComponent](#) of the entity being controlled in the first person view should be deleted once the game changes back to third person view.

Definition at line 112 of file `InputSystem.hpp`.

#### 6.50.4.5 `EntitySystem& InputSystem::entities_ [private]`

Reference to the game's entity system.

Definition at line 64 of file `InputSystem.hpp`.

#### 6.50.4.6 `bool InputSystem::first_person_ [private]`

Determines if the first person view is turned on.

Definition at line 69 of file `InputSystem.hpp`.

#### 6.50.4.7 `tdt::uint InputSystem::first_person_id_ [private]`

If the first person view is turned on, this holds ID of the entity being controlled.

Definition at line 74 of file `InputSystem.hpp`.

#### 6.50.4.8 `int InputSystem::KEY_UP [private]`

Current keybindings, allow rebinding.

Definition at line 84 of file `InputSystem.hpp`.

#### 6.50.4.9 `OIS::Keyboard& InputSystem::keyboard_ [private]`

Reference to the keyboard being used.

Definition at line 79 of file `InputSystem.hpp`.

#### 6.50.4.10 `std::unique_ptr<TaskHandlerComponent> InputSystem::task_backup_ [private]`

Backup of the task component when entering first person view.

Definition at line 106 of file `InputSystem.hpp`.

The documentation for this class was generated from the following files:

- `systems/InputSystem.hpp`
- `systems/InputSystem.cpp`

## 6.51 util::IS\_ENEMY Struct Reference

Tests if if the entity it is called on is an enemy of the entity specified in it's constructor.

```
#include <Util.hpp>
```

### Public Member Functions

- [IS\\_ENEMY](#) ([EntitySystem](#) &, tdt::uint)  
*Constructor.*
- [~IS\\_ENEMY](#) ()=default  
*Destructor.*
- bool [operator\(\)](#) (tdt::uint)  
*Tests if a given entity is an enemy of the entity specified in the constructor.*

### Private Attributes

- FACTION [enemy\\_faction\\_](#)  
*Faction that is hostile towards the entity performing the search.*
- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing all tested entities.*

#### 6.51.1 Detailed Description

Tests if if the entity it is called on is an enemy of the entity specified in it's constructor.

Definition at line 19 of file Util.hpp.

#### 6.51.2 Constructor & Destructor Documentation

##### 6.51.2.1 util::IS\_ENEMY::IS\_ENEMY ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Constructor.

##### Parameters

<i>Entity</i>	system containing all tested entities.
<i>ID</i>	of the entity towards which others are tested for the enemy status.

Definition at line 5 of file Util.cpp.

##### 6.51.2.2 util::IS\_ENEMY::~~IS\_ENEMY ( ) [default]

Destructor.



### 6.51.3 Member Function Documentation

#### 6.51.3.1 bool util::IS\_ENEMY::operator() ( tdt::uint *id* )

Tests if a given entity is an enemy of the entity specified in the constructor.

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 15 of file Util.cpp.

### 6.51.4 Member Data Documentation

#### 6.51.4.1 FACTION util::IS\_ENEMY::enemy\_faction\_ [private]

Faction that is hostile towards the entity performing the search.

Definition at line 45 of file Util.hpp.

#### 6.51.4.2 EntitySystem& util::IS\_ENEMY::entities\_ [private]

Entity system containing all tested entities.

Definition at line 50 of file Util.hpp.

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

- tools/Util.hpp
- tools/Util.cpp

## 6.52 util::IS\_FRIENDLY Struct Reference

Tests if the entity it is called on is a friend of the entity specified in it's constructor.

```
#include <Util.hpp>
```

### Public Member Functions

- [IS\\_FRIENDLY](#) (EntitySystem &, tdt::uint)  
*Constructor.*
- [~IS\\_FRIENDLY](#) ()=default  
*Destructor.*
- bool [operator\(\)](#) (tdt::uint)  
*Tests if a given entity is a friend of the entity specified in the constructor.*

## Private Attributes

- FACTION [faction\\_](#)  
*Faction that is friendly towards the entity performing the search.*
- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing all tested entities.*

### 6.52.1 Detailed Description

Tests if if the entity it is called on is a friend of the entity specified in it's constructor.

Definition at line 57 of file Util.hpp.

### 6.52.2 Constructor & Destructor Documentation

#### 6.52.2.1 `util::IS_FRIENDLY::IS_FRIENDLY ( EntitySystem & ents, tdt::uint id )`

Constructor.

Parameters

<i>Entity</i>	system containing all tested entities.
<i>ID</i>	of the entity towards which others are tested for the friendly status.

Definition at line 21 of file Util.cpp.

#### 6.52.2.2 `util::IS_FRIENDLY::~~IS_FRIENDLY ( )` `[default]`

Destructor.

### 6.52.3 Member Function Documentation

#### 6.52.3.1 `bool util::IS_FRIENDLY::operator() ( tdt::uint id )`

Tests if a given entity is a friend of the entity specified in the constructor.

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 25 of file Util.cpp.

### 6.52.4 Member Data Documentation

## 6.52.4.1 EntitySystem&amp; util::IS\_FRIENDLY::entities\_ [private]

Entity system containing all tested entities.

Definition at line 88 of file Util.hpp.

## 6.52.4.2 FACTION util::IS\_FRIENDLY::faction\_ [private]

Faction that is friendly towards the entity performing the search.

Definition at line 83 of file Util.hpp.

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

- tools/Util.hpp
- tools/Util.cpp

## 6.53 util::IS\_FRIENDLY\_OR\_NEUTRAL Struct Reference

Tests if the entity it is called on is a friend of or neutral to the entity specified in it's constructor.

```
#include <Util.hpp>
```

## Public Member Functions

- [IS\\_FRIENDLY\\_OR\\_NEUTRAL](#) ([EntitySystem](#) &, tdt::uint)  
*Constructor.*
- [~IS\\_FRIENDLY\\_OR\\_NEUTRAL](#) ()=default  
*Destructor.*
- bool [operator\(\)](#) (tdt::uint)  
*Tests if a given entity is a friend of or neutral to the entity specified in the constructor.*

## Private Attributes

- FACTION [faction\\_](#)  
*Faction that is friendly towards the entity performing the search.*
- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing all tested entities.*

## 6.53.1 Detailed Description

Tests if the entity it is called on is a friend of or neutral to the entity specified in it's constructor.

Definition at line 95 of file Util.hpp.

## 6.53.2 Constructor &amp; Destructor Documentation

6.53.2.1 util::IS\_FRIENDLY\_OR\_NEUTRAL::IS\_FRIENDLY\_OR\_NEUTRAL ( [EntitySystem](#) & *ents*, tdt::uint *id* )

Constructor.

## Parameters

<i>Entity</i>	system containing all tested entities.
<i>ID</i>	of the entity towards which others are tested for the friendly/neutrality status.

Definition at line 30 of file Util.cpp.

6.53.2.2 `util::IS_FRIENDLY_OR_NEUTRAL::~IS_FRIENDLY_OR_NEUTRAL ( ) [default]`

Destructor.

### 6.53.3 Member Function Documentation

6.53.3.1 `bool util::IS_FRIENDLY_OR_NEUTRAL::operator() ( tdt::uint id )`

Tests if a given entity is a friend of or neutral to the entity specified in the constructor.

## Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 34 of file Util.cpp.

### 6.53.4 Member Data Documentation

6.53.4.1 `EntitySystem& util::IS_FRIENDLY_OR_NEUTRAL::entities_ [private]`

Entity system containing all tested entities.

Definition at line 126 of file Util.hpp.

6.53.4.2 `FACTION util::IS_FRIENDLY_OR_NEUTRAL::faction_ [private]`

Faction that is friendly towards the entity performing the search.

Definition at line 121 of file Util.hpp.

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

- tools/Util.hpp
- tools/Util.cpp

## 6.54 util::IS\_GOLD\_VAULT Struct Reference

Tests if a given entity is of friendly faction, has structure component and has gold component (that is, it's a gold vault).

```
#include <Util.hpp>
```

### Public Member Functions

- [IS\\_GOLD\\_VAULT](#) ([EntitySystem](#) &)  
*Constructor.*
- [~IS\\_GOLD\\_VAULT](#) ()=default  
*Destructor.*
- bool [operator\(\)](#) (tdt::uint)  
*Tests a given entity.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system that contains all tested entities.*

#### 6.54.1 Detailed Description

Tests if a given entity is of friendly faction, has structure component and has gold component (that is, it's a gold vault).

Definition at line 163 of file Util.hpp.

#### 6.54.2 Constructor & Destructor Documentation

##### 6.54.2.1 util::IS\_GOLD\_VAULT::IS\_GOLD\_VAULT ( [EntitySystem](#) & *ents* )

Constructor.

Parameters

<i>Entity</i>	system that contains all tested entities.
---------------	---

Definition at line 110 of file Util.cpp.

##### 6.54.2.2 util::IS\_GOLD\_VAULT::~~IS\_GOLD\_VAULT ( ) [default]

Destructor.

### 6.54.3 Member Function Documentation

#### 6.54.3.1 bool util::IS\_GOLD\_VAULT::operator() ( tdt::uint *id* )

Tests a given entity.

Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 114 of file Util.cpp.

### 6.54.4 Member Data Documentation

#### 6.54.4.1 EntitySystem& util::IS\_GOLD\_VAULT::entities\_ [private]

Entity system that contains all tested entities.

Definition at line 186 of file Util.hpp.

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

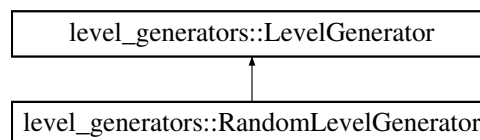
- tools/Util.hpp
- tools/Util.cpp

## 6.55 level\_generators::LevelGenerator Class Reference

Abstract parent class of all level generators, allows for different level generators used to create levels with minimal effort.

```
#include <LevelGenerators.hpp>
```

Inheritance diagram for level\_generators::LevelGenerator:



### Public Member Functions

- [LevelGenerator](#) ([EntitySystem](#) &, tdt::uint)  
*Constructor.*
- virtual [~LevelGenerator](#) ()=default  
*Destructor.*
- virtual void [generate](#) (tdt::uint, tdt::uint, [WaveSystem](#) &)=0  
*Generates a level with the given dimensions.*

## Protected Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system that contains the level's entities.*
- `tdt::uint` [cycles\\_](#)  
*Number of cycles done while generating the world.*

### 6.55.1 Detailed Description

Abstract parent class of all level generators, allows for different level generators used to create levels with minimal effort.

Definition at line 20 of file LevelGenerators.hpp.

### 6.55.2 Constructor & Destructor Documentation

#### 6.55.2.1 level\_generators::LevelGenerator::LevelGenerator ( EntitySystem & ents, tdt::uint c )

Constructor.

Parameters

<i>Entity</i>	system that contains the level's entities.
<i>Number</i>	of iterations done while generating.

Definition at line 10 of file LevelGenerators.cpp.

#### 6.55.2.2 virtual level\_generators::LevelGenerator::~~LevelGenerator ( ) [virtual],[default]

Destructor.

### 6.55.3 Member Function Documentation

#### 6.55.3.1 virtual void level\_generators::LevelGenerator::generate ( tdt::uint , tdt::uint , WaveSystem & ) [pure virtual]

Generates a level with the given dimensions.

Parameters

<i>Width</i>	of the level.
<i>Height</i>	of the level.
<i>Wave</i>	system that will have it's spawn nodes set.

Implemented in [level\\_generators::RandomLevelGenerator](#).

### 6.55.4 Member Data Documentation

#### 6.55.4.1 `tdt::uint level_generators::LevelGenerator::cycles_` [protected]

Number of cycles done while generating the world.

Definition at line 52 of file LevelGenerators.hpp.

#### 6.55.4.2 `EntitySystem& level_generators::LevelGenerator::entities_` [protected]

Entity system that contains the level's entities.

Definition at line 47 of file LevelGenerators.hpp.

The documentation for this class was generated from the following files:

- tools/LevelGenerators.hpp
- tools/LevelGenerators.cpp

## 6.56 LightComponent Struct Reference

Allows an entity to emit light to it's surrounding area.

```
#include <Components.hpp>
```

### Public Member Functions

- **LightComponent** (const [LightComponent](#) &)=default
- **LightComponent** ([LightComponent](#) &&)=default
- [LightComponent](#) & **operator=** (const [LightComponent](#) &)=default
- [LightComponent](#) & **operator=** ([LightComponent](#) &&)=default

### Public Attributes

- `Ogre::SceneNode * node`
- `Ogre::Light * light`

### Static Public Attributes

- static constexpr int **type** = 36

#### 6.56.1 Detailed Description

Allows an entity to emit light to it's surrounding area.

Definition at line 841 of file Components.hpp.

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

- Components.hpp



## 6.57 LimitedLifeSpanComponent Struct Reference

Allows to create entities that are automatically killed (summons) after a certain amount of time has passed (lifespan).

```
#include <Components.hpp>
```

### Public Member Functions

- **LimitedLifeSpanComponent** (tdt::real max=0.f)
- **LimitedLifeSpanComponent** (const [LimitedLifeSpanComponent](#) &)=default
- **LimitedLifeSpanComponent** ([LimitedLifeSpanComponent](#) &&)=default
- [LimitedLifeSpanComponent](#) & **operator=** (const [LimitedLifeSpanComponent](#) &)=default
- [LimitedLifeSpanComponent](#) & **operator=** ([LimitedLifeSpanComponent](#) &&)=default

### Public Attributes

- tdt::real **curr\_time**
- tdt::real **max\_time**

### Static Public Attributes

- static constexpr int **type** = 33

#### 6.57.1 Detailed Description

Allows to create entities that are automatically killed (summons) after a certain amount of time has passed (lifespan).

Definition at line 783 of file Components.hpp.

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

- Components.hpp

## 6.58 util::effect::LOWER\_SPEED\_EFFECT Struct Reference

Halves the speed of the entity it's called on for a given time period.

```
#include <Effects.hpp>
```

### Public Member Functions

- [LOWER\\_SPEED\\_EFFECT](#) ([EntitySystem](#) &, tdt::real)  
*Constructor.*
- [~LOWER\\_SPEED\\_EFFECT](#) ()=default  
*Destructor.*
- void [operator\(\)](#) (tdt::uint)  
*Halves the speed of a given entity.*

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing the entities this effect will be called on.*
- `tdt::real` [time\\_](#)  
*The time period that has to pass before the speed of the affected entities gets restored.*

### 6.58.1 Detailed Description

Halves the speed of the entity it's called on for a given time period.

Definition at line 93 of file Effects.hpp.

### 6.58.2 Constructor & Destructor Documentation

#### 6.58.2.1 `util::effect::LOWER_SPEED_EFFECT::LOWER_SPEED_EFFECT ( EntitySystem & ents, tdt::real time )`

Constructor.

##### Parameters

<i>Entity</i>	system containing the entities this effect will be called on.
<i>The</i>	time period before the speed is restored.

Definition at line 25 of file Effects.cpp.

#### 6.58.2.2 `util::effect::LOWER_SPEED_EFFECT::~~LOWER_SPEED_EFFECT ( )` `[default]`

Destructor.

### 6.58.3 Member Function Documentation

#### 6.58.3.1 `void util::effect::LOWER_SPEED_EFFECT::operator() ( tdt::uint id )`

Halves the speed of a given entity.

##### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 29 of file Effects.cpp.

### 6.58.4 Member Data Documentation

#### 6.58.4.1 EntitySystem& util::effect::LOWER\_SPEED\_EFFECT::entities\_ [private]

Entity system containing the entities this effect will be called on.

Definition at line 119 of file Effects.hpp.

#### 6.58.4.2 tdt::real util::effect::LOWER\_SPEED\_EFFECT::time\_ [private]

The time period that has to pass before the speed of the affected entities gets restored.

Definition at line 125 of file Effects.hpp.

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

- tools/Effects.hpp
- tools/Effects.cpp

## 6.59 LuaInterface Class Reference

Class that creates an interface between engine (C++) and logic (Lua) code.

```
#include <LuaInterface.hpp>
```

### Public Member Functions

- [LuaInterface](#) ()=delete  
*This is a static class, so all constructors are deleted.*
- [LuaInterface](#) (const [LuaInterface](#) &)=delete
- [LuaInterface](#) ([LuaInterface](#) &&)=delete

### Static Public Member Functions

- static void [init](#) ([Game](#) \*)  
*Sets the lua\_this pointer and registers all C++ API functions to Lua.*

## Static Private Member Functions

- static int [lua\\_get\\_avg\\_fps](#) (lpp::Script::state)
- static int [lua\\_get\\_fps](#) (lpp::Script::state)
- static int [lua\\_print](#) (lpp::Script::state)
- static int [lua\\_set\\_game\\_state](#) (lpp::Script::state)
- static int [lua\\_toggle\\_bounding\\_boxes](#) (lpp::Script::state)
- static int [lua\\_toggle\\_camera\\_free\\_mode](#) (lpp::Script::state)
- static int [lua\\_toggle\\_entity\\_creator](#) (lpp::Script::state)
- static int [lua\\_list\\_selected](#) (lpp::Script::state)
- static int [lua\\_destroy\\_selected](#) (lpp::Script::state)
- static int [lua\\_kill\\_selected](#) (lpp::Script::state)
- static int [lua\\_list\\_components\\_of](#) (lpp::Script::state)
- static int [lua\\_load](#) (lpp::Script::state)
- static int [lua\\_reload\\_all](#) (lpp::Script::state)
- static int [lua\\_save\\_game](#) (lpp::Script::state)
- static int [lua\\_load\\_game](#) (lpp::Script::state)
- static int [lua\\_get\\_cursor\\_position](#) (lpp::Script::state)
- static int [lua\\_can\\_place\\_when\\_game\\_paused](#) (lpp::Script::state)
- static int [lua\\_toggle\\_placing\\_when\\_game\\_paused](#) (lpp::Script::state)
- static int [lua\\_new\\_game](#) (lpp::Script::state)
- static int [lua\\_create\\_empty\\_level](#) (lpp::Script::state)
- static int [lua\\_reset\\_unlocks](#) (lpp::Script::state)
- static int [lua\\_get\\_random](#) (lpp::Script::state)
- static int [lua\\_set\\_key\\_bind](#) (lpp::Script::state)
- static int [lua\\_get\\_first\\_selected](#) (lpp::Script::state)
- static int [lua\\_get\\_enemies](#) (lpp::Script::state)
- static int [lua\\_get\\_friends](#) (lpp::Script::state)
- static int [lua\\_set\\_throne\\_id](#) (lpp::Script::state)
- static int [lua\\_get\\_throne\\_id](#) (lpp::Script::state)
- static int [lua\\_command\\_to\\_mine](#) (lpp::Script::state)
- static int [lua\\_command\\_to\\_attack](#) (lpp::Script::state)
- static int [lua\\_command\\_to\\_reposition](#) (lpp::Script::state)
- static int [lua\\_command\\_to\\_return\\_gold](#) (lpp::Script::state)
- static int [lua\\_command\\_to\\_fall\\_back](#) (lpp::Script::state)
- static int [lua\\_get\\_enum\\_direction](#) (lpp::Script::state)
- static int [lua\\_get\\_node\\_in\\_dir](#) (lpp::Script::state)
- static int [lua\\_set\\_mesh](#) (lpp::Script::state)
- static int [lua\\_set\\_material](#) (lpp::Script::state)
- static int [lua\\_set\\_visible](#) (lpp::Script::state)
- static int [lua\\_set\\_manual\\_scaling](#) (lpp::Script::state)
- static int [lua\\_set\\_scale](#) (lpp::Script::state)
- static int [lua\\_get\\_mesh](#) (lpp::Script::state)
- static int [lua\\_get\\_material](#) (lpp::Script::state)
- static int [lua\\_is\\_visible](#) (lpp::Script::state)
- static int [lua\\_get\\_manual\\_scaling](#) (lpp::Script::state)
- static int [lua\\_get\\_scale](#) (lpp::Script::state)
- static int [lua\\_look\\_at](#) (lpp::Script::state)
- static int [lua\\_rotate\\_x](#) (lpp::Script::state)
- static int [lua\\_rotate\\_y](#) (lpp::Script::state)
- static int [lua\\_rotate\\_z](#) (lpp::Script::state)
- static int [lua\\_collide](#) (lpp::Script::state)
- static int [lua\\_set\\_query\\_flags](#) (lpp::Script::state)
- static int [lua\\_get\\_query\\_flags](#) (lpp::Script::state)
- static int [lua\\_apply\\_scale](#) (lpp::Script::state)

- static int **lua\_set\_graphics\_update\_period** (lpp::Script::state)
- static int **lua\_get\_graphics\_update\_period** (lpp::Script::state)
- static int **lua\_create\_entity** (lpp::Script::state)
- static int **lua\_destroy\_entity** (lpp::Script::state)
- static int **lua\_add\_component** (lpp::Script::state)
- static int **lua\_delete\_component** (lpp::Script::state)
- static int **lua\_init\_graphics\_component** (lpp::Script::state)
- static int **lua\_list\_entity\_tables** (lpp::Script::state)
- static int **lua\_place\_entity** (lpp::Script::state)
- static int **lua\_register\_entity** (lpp::Script::state)
- static int **lua\_exists** (lpp::Script::state)
- static int **lua\_kill\_entity** (lpp::Script::state)
- static int **lua\_has\_component** (lpp::Script::state)
- static int **lua\_entity\_reset\_state** (lpp::Script::state)
- static int **lua\_set\_position** (lpp::Script::state)
- static int **lua\_get\_position** (lpp::Script::state)
- static int **lua\_is\_solid** (lpp::Script::state)
- static int **lua\_set\_solid** (lpp::Script::state)
- static int **lua\_set\_half\_height** (lpp::Script::state)
- static int **lua\_get\_half\_height** (lpp::Script::state)
- static int **lua\_get\_distance** (lpp::Script::state)
- static int **lua\_get\_angle** (lpp::Script::state)
- static int **lua\_get\_angle\_between** (lpp::Script::state)
- static int **lua\_set\_2d\_position** (lpp::Script::state)
- static int **lua\_get\_2d\_position** (lpp::Script::state)
- static int **lua\_move\_to** (lpp::Script::state)
- static int **lua\_move** (lpp::Script::state)
- static int **lua\_can\_move\_to** (lpp::Script::state)
- static int **lua\_get\_speed\_modifier** (lpp::Script::state)
- static int **lua\_set\_speed\_modifier** (lpp::Script::state)
- static int **lua\_dir\_to** (lpp::Script::state)
- static int **lua\_get\_dir** (lpp::Script::state)
- static int **lua\_get\_dir\_back** (lpp::Script::state)
- static int **lua\_get\_dir\_left** (lpp::Script::state)
- static int **lua\_get\_dir\_right** (lpp::Script::state)
- static int **lua\_set\_original\_speed** (lpp::Script::state)
- static int **lua\_get\_original\_speed** (lpp::Script::state)
- static int **lua\_reset\_speed** (lpp::Script::state)
- static int **lua\_set\_health** (lpp::Script::state)
- static int **lua\_get\_health** (lpp::Script::state)
- static int **lua\_add\_health** (lpp::Script::state)
- static int **lua\_sub\_health** (lpp::Script::state)
- static int **lua\_heal** (lpp::Script::state)
- static int **lua\_buff** (lpp::Script::state)
- static int **lua\_debuff** (lpp::Script::state)
- static int **lua\_set\_defense** (lpp::Script::state)
- static int **lua\_get\_defense** (lpp::Script::state)
- static int **lua\_add\_defense** (lpp::Script::state)
- static int **lua\_sub\_defense** (lpp::Script::state)
- static int **lua\_set\_regen** (lpp::Script::state)
- static int **lua\_get\_regen** (lpp::Script::state)
- static int **lua\_set\_alive** (lpp::Script::state)
- static int **lua\_is\_alive** (lpp::Script::state)
- static int **lua\_ubercharge** (lpp::Script::state)
- static int **lua\_set\_regen\_period** (lpp::Script::state)

- static int **lua\_get\_regen\_period** (lpp::Script::state)
- static int **lua\_get\_blueprint** (lpp::Script::state)
- static int **lua\_get\_state** (lpp::Script::state)
- static int **lua\_get\_faction** (lpp::Script::state)
- static int **lua\_set\_blueprint** (lpp::Script::state)
- static int **lua\_set\_state** (lpp::Script::state)
- static int **lua\_set\_faction** (lpp::Script::state)
- static int **lua\_set\_update\_period** (lpp::Script::state)
- static int **lua\_get\_update\_period** (lpp::Script::state)
- static int **lua\_force\_update** (lpp::Script::state)
- static int **lua\_get\_faction\_name** (lpp::Script::state)
- static int **lua\_set\_input\_handler** (lpp::Script::state)
- static int **lua\_get\_input\_handler** (lpp::Script::state)
- static int **lua\_toggle\_first\_person** (lpp::Script::state)
- static int **lua\_add\_node** (lpp::Script::state)
- static int **lua\_get\_node** (lpp::Script::state)
- static int **lua\_get\_node\_from\_position** (lpp::Script::state)
- static int **lua\_create\_grid\_graphics** (lpp::Script::state)
- static int **lua\_delete\_grid\_graphics** (lpp::Script::state)
- static int **lua\_toggle\_grid\_visible** (lpp::Script::state)
- static int **lua\_is\_free** (lpp::Script::state)
- static int **lua\_set\_free** (lpp::Script::state)
- static int **lua\_set\_free\_selected** (lpp::Script::state)
- static int **lua\_pathfind** (lpp::Script::state)
- static int **lua\_pop\_first\_path\_node** (lpp::Script::state)
- static int **lua\_pop\_last\_path\_node** (lpp::Script::state)
- static int **lua\_path\_queue\_empty** (lpp::Script::state)
- static int **lua\_clear\_path** (lpp::Script::state)
- static int **lua\_set\_pathfinding\_blueprint** (lpp::Script::state)
- static int **lua\_get\_pathfinding\_blueprint** (lpp::Script::state)
- static int **lua\_create\_graph** (lpp::Script::state)
- static int **lua\_set\_resident** (lpp::Script::state)
- static int **lua\_get\_resident** (lpp::Script::state)
- static int **lua\_add\_residences** (lpp::Script::state)
- static int **lua\_add\_residence** (lpp::Script::state)
- static int **lua\_set\_radius** (lpp::Script::state)
- static int **lua\_set\_walk\_through** (lpp::Script::state)
- static int **lua\_is\_walk\_throuth** (lpp::Script::state)
- static int **lua\_place\_at\_random\_free\_node** (lpp::Script::state)
- static int **lua\_distribute\_to\_adjacent\_free\_nodes** (lpp::Script::state)
- static int **lua\_get\_random\_free\_node** (lpp::Script::state)
- static int **lua\_set\_portal\_neighbour** (lpp::Script::state)
- static int **lua\_get\_next\_pathfinding\_node** (lpp::Script::state)
- static int **lua\_get\_target\_pathfinding\_node** (lpp::Script::state)
- static int **lua\_pathfinding\_skip\_next\_node** (lpp::Script::state)
- static int **lua\_pathfinding\_after\_next\_node** (lpp::Script::state)
- static int **lua\_add\_task** (lpp::Script::state)
- static int **lua\_add\_priority\_task** (lpp::Script::state)
- static int **lua\_cancel\_task** (lpp::Script::state)
- static int **lua\_create\_task** (lpp::Script::state)
- static int **lua\_list\_tasks\_of** (lpp::Script::state)
- static int **lua\_task\_possible** (lpp::Script::state)
- static int **lua\_task\_type\_possibe** (lpp::Script::state)
- static int **lua\_clear\_task\_queue** (lpp::Script::state)
- static int **lua\_set\_task\_source** (lpp::Script::state)

- static int **lua\_get\_task\_source** (lpp::Script::state)
- static int **lua\_set\_task\_target** (lpp::Script::state)
- static int **lua\_get\_task\_target** (lpp::Script::state)
- static int **lua\_set\_task\_type** (lpp::Script::state)
- static int **lua\_get\_task\_type** (lpp::Script::state)
- static int **lua\_add\_possible\_task** (lpp::Script::state)
- static int **lua\_delete\_possible\_task** (lpp::Script::state)
- static int **lua\_set\_task\_handling\_blueprint** (lpp::Script::state)
- static int **lua\_get\_task\_handling\_blueprint** (lpp::Script::state)
- static int **lua\_set\_task\_complete** (lpp::Script::state)
- static int **lua\_is\_task\_complete** (lpp::Script::state)
- static int **lua\_task\_clear** (lpp::Script::state)
- static int **lua\_set\_combat\_target** (lpp::Script::state)
- static int **lua\_get\_combat\_target** (lpp::Script::state)
- static int **lua\_set\_range** (lpp::Script::state)
- static int **lua\_get\_range** (lpp::Script::state)
- static int **lua\_set\_dmg\_range** (lpp::Script::state)
- static int **lua\_get\_dmg\_range** (lpp::Script::state)
- static int **lua\_get\_dmg** (lpp::Script::state)
- static int **lua\_set\_cooldown** (lpp::Script::state)
- static int **lua\_get\_cooldown** (lpp::Script::state)
- static int **lua\_set\_atk\_type** (lpp::Script::state)
- static int **lua\_get\_atk\_type** (lpp::Script::state)
- static int **lua\_set\_homing\_source** (lpp::Script::state)
- static int **lua\_get\_homing\_source** (lpp::Script::state)
- static int **lua\_set\_homing\_target** (lpp::Script::state)
- static int **lua\_get\_homing\_target** (lpp::Script::state)
- static int **lua\_set\_homing\_dmg** (lpp::Script::state)
- static int **lua\_get\_homing\_dmg** (lpp::Script::state)
- static int **lua\_closest\_enemy\_in\_sight** (lpp::Script::state)
- static int **lua\_closest\_friendly\_in\_sight** (lpp::Script::state)
- static int **lua\_closest\_enemy** (lpp::Script::state)
- static int **lua\_closest\_friendly** (lpp::Script::state)
- static int **lua\_closest\_enemy\_in\_sight\_thats\_not** (lpp::Script::state)
- static int **lua\_closest\_friendly\_in\_sight\_thats\_not** (lpp::Script::state)
- static int **lua\_closest\_enemy\_thats\_not** (lpp::Script::state)
- static int **lua\_closest\_friendly\_thats\_not** (lpp::Script::state)
- static int **lua\_in\_sight** (lpp::Script::state)
- static int **lua\_run\_away\_from** (lpp::Script::state)
- static int **lua\_set\_max\_run\_away\_attempts** (lpp::Script::state)
- static int **lua\_get\_max\_run\_away\_attempts** (lpp::Script::state)
- static int **lua\_apply\_heal\_to\_entities\_in\_range** (lpp::Script::state)
- static int **lua\_apply\_damage\_to\_entities\_in\_range** (lpp::Script::state)
- static int **lua\_apply\_slow\_to\_entities\_in\_range** (lpp::Script::state)
- static int **lua\_apply\_freeze\_to\_entities\_in\_range** (lpp::Script::state)
- static int **lua\_in\_range** (lpp::Script::state)
- static int **lua\_set\_projectile\_blueprint** (lpp::Script::state)
- static int **lua\_get\_projectile\_blueprint** (lpp::Script::state)
- static int **lua\_apply\_slow\_to** (lpp::Script::state)
- static int **lua\_apply\_freeze\_to** (lpp::Script::state)
- static int **lua\_enemy\_in\_range** (lpp::Script::state)
- static int **lua\_closest\_friendly\_structure** (lpp::Script::state)
- static int **lua\_closest\_enemy\_structure** (lpp::Script::state)
- static int **lua\_closest\_friendly\_structure\_in\_sight** (lpp::Script::state)
- static int **lua\_closest\_enemy\_structure\_in\_sight** (lpp::Script::state)

- static int **lua\_set\_production\_blueprint** (lpp::Script::state)
- static int **lua\_get\_production\_blueprint** (lpp::Script::state)
- static int **lua\_set\_production\_limit** (lpp::Script::state)
- static int **lua\_get\_production\_limit** (lpp::Script::state)
- static int **lua\_set\_production\_cooldown** (lpp::Script::state)
- static int **lua\_get\_production\_cooldown** (lpp::Script::state)
- static int **lua\_set\_production\_progress** (lpp::Script::state)
- static int **lua\_get\_production\_progress** (lpp::Script::state)
- static int **lua\_set\_production\_count** (lpp::Script::state)
- static int **lua\_get\_production\_count** (lpp::Script::state)
- static int **lua\_set\_producer** (lpp::Script::state)
- static int **lua\_get\_producer** (lpp::Script::state)
- static int **lua\_instant\_production** (lpp::Script::state)
- static int **lua\_set\_production\_multiplier** (lpp::Script::state)
- static int **lua\_get\_production\_multiplier** (lpp::Script::state)
- static int **lua\_double\_production** (lpp::Script::state)
- static int **lua\_increase\_production** (lpp::Script::state)
- static int **lua\_get\_curr\_time** (lpp::Script::state)
- static int **lua\_advance\_curr\_time** (lpp::Script::state)
- static int **lua\_max\_curr\_time** (lpp::Script::state)
- static int **lua\_set\_time\_limit** (lpp::Script::state)
- static int **lua\_get\_time\_limit** (lpp::Script::state)
- static int **lua\_set\_timer\_target** (lpp::Script::state)
- static int **lua\_get\_timer\_target** (lpp::Script::state)
- static int **lua\_set\_timer\_type** (lpp::Script::state)
- static int **lua\_get\_timer\_type** (lpp::Script::state)
- static int **lua\_advance\_all\_timers** (lpp::Script::state)
- static int **lua\_advance\_all\_timers\_of\_type** (lpp::Script::state)
- static int **lua\_set\_timer\_multiplier** (lpp::Script::state)
- static int **lua\_get\_timer\_multiplier** (lpp::Script::state)
- static int **lua\_set\_event\_type** (lpp::Script::state)
- static int **lua\_get\_event\_type** (lpp::Script::state)
- static int **lua\_set\_event\_target** (lpp::Script::state)
- static int **lua\_get\_event\_target** (lpp::Script::state)
- static int **lua\_set\_event\_radius** (lpp::Script::state)
- static int **lua\_get\_event\_radius** (lpp::Script::state)
- static int **lua\_set\_event\_active** (lpp::Script::state)
- static int **lua\_is\_event\_active** (lpp::Script::state)
- static int **lua\_set\_handler\_of\_event** (lpp::Script::state)
- static int **lua\_get\_handler\_of\_event** (lpp::Script::state)
- static int **lua\_set\_event\_handler** (lpp::Script::state)
- static int **lua\_get\_event\_handler** (lpp::Script::state)
- static int **lua\_can\_handle\_event** (lpp::Script::state)
- static int **lua\_add\_possible\_event** (lpp::Script::state)
- static int **lua\_delete\_possible\_event** (lpp::Script::state)
- static int **lua\_set\_event\_update\_period** (lpp::Script::state)
- static int **lua\_get\_event\_update\_period** (lpp::Script::state)
- static int **lua\_set\_event\_update\_multiplier** (lpp::Script::state)
- static int **lua\_get\_event\_update\_multiplier** (lpp::Script::state)
- static int **lua\_set\_destructor\_blueprint** (lpp::Script::state)
- static int **lua\_get\_destructor\_blueprint** (lpp::Script::state)
- static int **lua\_set\_curr\_gold** (lpp::Script::state)
- static int **lua\_get\_curr\_gold** (lpp::Script::state)
- static int **lua\_set\_max\_gold** (lpp::Script::state)
- static int **lua\_get\_max\_gold** (lpp::Script::state)



- static int **lua\_add\_gold** (lpp::Script::state)
- static int **lua\_sub\_gold** (lpp::Script::state)
- static int **lua\_transfer\_all\_gold** (lpp::Script::state)
- static int **lua\_get\_closest\_gold\_deposit** (lpp::Script::state)
- static int **lua\_get\_closest\_gold\_deposit\_in\_sight** (lpp::Script::state)
- static int **lua\_gold\_full** (lpp::Script::state)
- static int **lua\_is\_gold\_vault** (lpp::Script::state)
- static int **lua\_get\_closest\_gold\_vault** (lpp::Script::state)
- static int **lua\_get\_closest\_gold\_vault\_in\_sight** (lpp::Script::state)
- static int **lua\_get\_closest\_free\_gold\_vault** (lpp::Script::state)
- static int **lua\_get\_closest\_free\_gold\_vault\_in\_sight** (lpp::Script::state)
- static int **lua\_exists\_free\_gold\_vault** (lpp::Script::state)
- static int **lua\_set\_gui\_visible** (lpp::Script::state)
- static int **lua\_is\_gui\_visible** (lpp::Script::state)
- static int **lua\_set\_window\_visible** (lpp::Script::state)
- static int **lua\_is\_window\_visible** (lpp::Script::state)
- static int **lua\_show\_save\_dialog** (lpp::Script::state)
- static int **lua\_show\_load\_dialog** (lpp::Script::state)
- static int **lua\_clear\_log** (lpp::Script::state)
- static int **lua\_print\_to\_log** (lpp::Script::state)
- static int **lua\_set\_log\_history** (lpp::Script::state)
- static int **lua\_get\_log\_history** (lpp::Script::state)
- static int **lua\_set\_log\_visible** (lpp::Script::state)
- static int **lua\_is\_log\_visible** (lpp::Script::state)
- static int **lua\_set\_tracked\_entity** (lpp::Script::state)
- static int **lua\_get\_tracked\_entity** (lpp::Script::state)
- static int **lua\_update\_tracking** (lpp::Script::state)
- static int **lua\_clear\_entity\_tracker** (lpp::Script::state)
- static int **lua\_set\_tracker\_visible** (lpp::Script::state)
- static int **lua\_is\_tracker\_visible** (lpp::Script::state)
- static int **lua\_console\_scroll\_down** (lpp::Script::state)
- static int **lua\_set\_console\_history** (lpp::Script::state)
- static int **lua\_get\_console\_history** (lpp::Script::state)
- static int **lua\_set\_console\_visible** (lpp::Script::state)
- static int **lua\_is\_console\_visible** (lpp::Script::state)
- static int **lua\_clear\_console** (lpp::Script::state)
- static int **lua\_set\_builder\_visible** (lpp::Script::state)
- static int **lua\_is\_builder\_visible** (lpp::Script::state)
- static int **lua\_register\_building** (lpp::Script::state)
- static int **lua\_research\_show** (lpp::Script::state)
- static int **lua\_free\_research** (lpp::Script::state)
- static int **lua\_research\_all** (lpp::Script::state)
- static int **lua\_dummy\_unlock** (lpp::Script::state)
- static int **lua\_research\_reset** (lpp::Script::state)
- static int **lua\_add\_player\_gold** (lpp::Script::state)
- static int **lua\_sub\_player\_gold** (lpp::Script::state)
- static int **lua\_add\_player\_mana** (lpp::Script::state)
- static int **lua\_sub\_player\_mana** (lpp::Script::state)
- static int **lua\_add\_player\_max\_units** (lpp::Script::state)
- static int **lua\_sub\_player\_max\_units** (lpp::Script::state)
- static int **lua\_add\_player\_curr\_units** (lpp::Script::state)
- static int **lua\_sub\_player\_curr\_units** (lpp::Script::state)
- static int **lua\_get\_player\_gold** (lpp::Script::state)
- static int **lua\_get\_player\_mana** (lpp::Script::state)
- static int **lua\_player\_reset** (lpp::Script::state)

- static int **lua\_nulify\_player\_stats** (lpp::Script::state)
- static int **lua\_add\_player\_max\_mana** (lpp::Script::state)
- static int **lua\_sub\_player\_max\_mana** (lpp::Script::state)
- static int **lua\_get\_player\_max\_mana** (lpp::Script::state)
- static int **lua\_add\_player\_mana\_regen** (lpp::Script::state)
- static int **lua\_sub\_player\_mana\_regen** (lpp::Script::state)
- static int **lua\_get\_player\_mana\_regen** (lpp::Script::state)
- static int **lua\_set\_price** (lpp::Script::state)
- static int **lua\_get\_price** (lpp::Script::state)
- static int **lua\_register\_spell** (lpp::Script::state)
- static int **lua\_spellcaster\_set\_type** (lpp::Script::state)
- static int **lua\_spellcaster\_get\_type** (lpp::Script::state)
- static int **lua\_spellcaster\_set\_spell** (lpp::Script::state)
- static int **lua\_spellcaster\_get\_spell** (lpp::Script::state)
- static int **lua\_spellcaster\_get\_last\_type** (lpp::Script::state)
- static int **lua\_spellcaster\_get\_last\_spell** (lpp::Script::state)
- static int **lua\_spellcaster\_set\_last\_spell\_id** (lpp::Script::state)
- static int **lua\_spellcaster\_get\_last\_spell\_id** (lpp::Script::state)
- static int **lua\_spellcaster\_is\_casting** (lpp::Script::state)
- static int **lua\_spellcaster\_stop\_casting** (lpp::Script::state)
- static int **lua\_align\_set\_material** (lpp::Script::state)
- static int **lua\_align\_get\_material** (lpp::Script::state)
- static int **lua\_align\_set\_mesh** (lpp::Script::state)
- static int **lua\_align\_get\_mesh** (lpp::Script::state)
- static int **lua\_align\_set\_position\_offset** (lpp::Script::state)
- static int **lua\_align\_get\_position\_offset** (lpp::Script::state)
- static int **lua\_align\_set\_scale** (lpp::Script::state)
- static int **lua\_align\_get\_scale** (lpp::Script::state)
- static int **lua\_mana\_crystal\_set\_cap** (lpp::Script::state)
- static int **lua\_mana\_crystal\_get\_cap** (lpp::Script::state)
- static int **lua\_mana\_crystal\_set\_regen** (lpp::Script::state)
- static int **lua\_mana\_crystal\_get\_regen** (lpp::Script::state)
- static int **lua\_on\_hit\_set\_blueprint** (lpp::Script::state)
- static int **lua\_on\_hit\_get\_blueprint** (lpp::Script::state)
- static int **lua\_on\_hit\_call** (lpp::Script::state)
- static int **lua\_on\_hit\_set\_cooldown** (lpp::Script::state)
- static int **lua\_on\_hit\_get\_cooldown** (lpp::Script::state)
- static int **lua\_constructor\_set\_blueprint** (lpp::Script::state)
- static int **lua\_constructor\_get\_blueprint** (lpp::Script::state)
- static int **lua\_constructor\_call** (lpp::Script::state)
- static int **lua\_trigger\_set\_blueprint** (lpp::Script::state)
- static int **lua\_trigger\_get\_blueprint** (lpp::Script::state)
- static int **lua\_trigger\_set\_linked\_entity** (lpp::Script::state)
- static int **lua\_trigger\_get\_linked\_entity** (lpp::Script::state)
- static int **lua\_trigger\_set\_cooldown** (lpp::Script::state)
- static int **lua\_trigger\_get\_cooldown** (lpp::Script::state)
- static int **lua\_trigger\_trigger** (lpp::Script::state)
- static int **lua\_trigger\_set\_check\_period** (lpp::Script::state)
- static int **lua\_trigger\_get\_check\_period** (lpp::Script::state)
- static int **lua\_trigger\_can\_be\_triggered\_by** (lpp::Script::state)
- static int **lua\_trigger\_reset\_timer** (lpp::Script::state)
- static int **lua\_trigger\_set\_radius** (lpp::Script::state)
- static int **lua\_trigger\_get\_radius** (lpp::Script::state)
- static int **lua\_upgrade\_set\_blueprint** (lpp::Script::state)
- static int **lua\_upgrade\_get\_blueprint** (lpp::Script::state)

- static int **lua\_upgrade\_set\_experience** (lpp::Script::state)
- static int **lua\_upgrade\_get\_experience** (lpp::Script::state)
- static int **lua\_upgrade\_add\_experience** (lpp::Script::state)
- static int **lua\_upgrade\_set\_exp\_needed** (lpp::Script::state)
- static int **lua\_upgrade\_get\_exp\_needed** (lpp::Script::state)
- static int **lua\_upgrade\_set\_level** (lpp::Script::state)
- static int **lua\_upgrade\_get\_level** (lpp::Script::state)
- static int **lua\_upgrade\_set\_level\_cap** (lpp::Script::state)
- static int **lua\_upgrade\_get\_level\_cap** (lpp::Script::state)
- static int **lua\_upgrade\_can\_level\_up** (lpp::Script::state)
- static int **lua\_upgrade\_upgrade** (lpp::Script::state)
- static int **lua\_upgrade\_all\_level\_up** (lpp::Script::state)
- static int **lua\_notification\_set\_cooldown** (lpp::Script::state)
- static int **lua\_notification\_get\_cooldown** (lpp::Script::state)
- static int **lua\_notification\_reset** (lpp::Script::state)
- static int **lua\_notification\_notify** (lpp::Script::state)
- static int **lua\_notification\_get\_curr\_time** (lpp::Script::state)
- static int **lua\_notification\_advance\_curr\_time** (lpp::Script::state)
- static int **lua\_explosion\_set\_delta** (lpp::Script::state)
- static int **lua\_explosion\_get\_delta** (lpp::Script::state)
- static int **lua\_explosion\_set\_max\_radius** (lpp::Script::state)
- static int **lua\_explosion\_get\_max\_radius** (lpp::Script::state)
- static int **lua\_explosion\_get\_curr\_radius** (lpp::Script::state)
- static int **lua\_explosion\_increase\_curr\_radius** (lpp::Script::state)
- static int **lua\_lls\_set\_max\_time** (lpp::Script::state)
- static int **lua\_lls\_get\_max\_time** (lpp::Script::state)
- static int **lua\_lls\_get\_curr\_time** (lpp::Script::state)
- static int **lua\_lls\_advance\_curr\_time** (lpp::Script::state)
- static int **lua\_name\_set** (lpp::Script::state)
- static int **lua\_name\_get** (lpp::Script::state)
- static int **lua\_exp\_val\_set** (lpp::Script::state)
- static int **lua\_exp\_val\_get** (lpp::Script::state)
- static int **lua\_exp\_val\_inc** (lpp::Script::state)
- static int **lua\_exp\_val\_dec** (lpp::Script::state)
- static int **lua\_mana\_set\_regen\_period** (lpp::Script::state)
- static int **lua\_mana\_get\_regen\_period** (lpp::Script::state)
- static int **lua\_wave\_next\_wave** (lpp::Script::state)
- static int **lua\_wave\_advance\_countdown** (lpp::Script::state)
- static int **lua\_wave\_entity\_died** (lpp::Script::state)
- static int **lua\_wave\_start** (lpp::Script::state)
- static int **lua\_wave\_pause** (lpp::Script::state)
- static int **lua\_wave\_set\_entity\_total** (lpp::Script::state)
- static int **lua\_wave\_get\_entity\_total** (lpp::Script::state)
- static int **lua\_wave\_set\_wave\_count** (lpp::Script::state)
- static int **lua\_wave\_get\_wave\_count** (lpp::Script::state)
- static int **lua\_wave\_add\_spawn\_node** (lpp::Script::state)
- static int **lua\_wave\_clear\_spawn\_nodes** (lpp::Script::state)
- static int **lua\_wave\_set\_spawn\_cooldown** (lpp::Script::state)
- static int **lua\_wave\_get\_spawn\_cooldown** (lpp::Script::state)
- static int **lua\_wave\_add\_entity\_blueprint** (lpp::Script::state)
- static int **lua\_wave\_set\_table** (lpp::Script::state)
- static int **lua\_wave\_get\_table** (lpp::Script::state)
- static int **lua\_wave\_set\_curr\_wave\_number** (lpp::Script::state)
- static int **lua\_wave\_get\_curr\_wave\_number** (lpp::Script::state)
- static int **lua\_wave\_set\_countdown** (lpp::Script::state)

- static int **lua\_wave\_get\_countdown** (lpp::Script::state)
- static int **lua\_wave\_set\_state** (lpp::Script::state)
- static int **lua\_wave\_get\_state** (lpp::Script::state)
- static int **lua\_wave\_update\_label\_text** (lpp::Script::state)
- static int **lua\_wave\_set\_spawn\_timer** (lpp::Script::state)
- static int **lua\_wave\_get\_spawn\_timer** (lpp::Script::state)
- static int **lua\_wave\_set\_wave\_entities** (lpp::Script::state)
- static int **lua\_wave\_get\_wave\_entities** (lpp::Script::state)
- static int **lua\_wave\_set\_entities\_spawned** (lpp::Script::state)
- static int **lua\_wave\_get\_entities\_spawned** (lpp::Script::state)
- static int **lua\_wave\_clear\_entity\_blueprints** (lpp::Script::state)
- static int **lua\_wave\_list** (lpp::Script::state)
- static int **lua\_wave\_set\_endless\_mode** (lpp::Script::state)
- static int **lua\_wave\_get\_endless\_mode** (lpp::Script::state)
- static int **lua\_wave\_turn\_endless\_on** (lpp::Script::state)
- static int **lua\_msg\_to\_plr\_show** (lpp::Script::state)
- static int **lua\_msg\_to\_plr\_show\_ok** (lpp::Script::state)
- static int **lua\_msg\_to\_plr\_show\_yes\_no** (lpp::Script::state)
- static int **lua\_msg\_set\_butl\_label** (lpp::Script::state)
- static int **lua\_msg\_reset\_butl\_labels** (lpp::Script::state)
- static int **lua\_mana\_add** (lpp::Script::state)
- static int **lua\_mana\_sub** (lpp::Script::state)
- static int **lua\_mana\_set** (lpp::Script::state)
- static int **lua\_mana\_get** (lpp::Script::state)
- static int **lua\_mana\_set\_max** (lpp::Script::state)
- static int **lua\_mana\_get\_max** (lpp::Script::state)
- static int **lua\_mana\_set\_regen** (lpp::Script::state)
- static int **lua\_mana\_get\_regen** (lpp::Script::state)
- static int **lua\_ent\_spell\_set\_blueprint** (lpp::Script::state)
- static int **lua\_ent\_spell\_get\_blueprint** (lpp::Script::state)
- static int **lua\_ent\_spell\_set\_cooldown** (lpp::Script::state)
- static int **lua\_ent\_spell\_get\_cooldown** (lpp::Script::state)
- static int **lua\_ent\_spell\_advance\_curr\_time** (lpp::Script::state)
- static int **lua\_ent\_spell\_set\_curr\_time** (lpp::Script::state)
- static int **lua\_ent\_spell\_get\_curr\_time** (lpp::Script::state)
- static int **lua\_ent\_spell\_cast** (lpp::Script::state)
- static int **lua\_light\_set\_visible** (lpp::Script::state)
- static int **lua\_light\_toggle\_visible** (lpp::Script::state)
- static int **lua\_light\_is\_visible** (lpp::Script::state)
- static int **lua\_light\_init** (lpp::Script::state)
- static int **lua\_command\_set** (lpp::Script::state)
- static int **lua\_command\_test** (lpp::Script::state)
- static int **lua\_counter\_increment** (lpp::Script::state)
- static int **lua\_counter\_decrement** (lpp::Script::state)
- static int **lua\_counter\_set\_curr\_value** (lpp::Script::state)
- static int **lua\_counter\_get\_curr\_value** (lpp::Script::state)
- static int **lua\_counter\_set\_max\_value** (lpp::Script::state)
- static int **lua\_counter\_get\_max\_value** (lpp::Script::state)

### Static Private Attributes

- static [Game](#) \* **lua\_this** {}  
*Lua requires all functions registered in it to be static (because it doesn't know anything about C++).*
- static [EntitySystem](#) \* **ents** {}

### 6.59.1 Detailed Description

Class that creates an interface between engine (C++) and logic (Lua) code.

Definition at line 12 of file LuaInterface.hpp.

### 6.59.2 Constructor & Destructor Documentation

#### 6.59.2.1 LuaInterface::LuaInterface ( ) [delete]

This is a static class, so all constructors are deleted.

### 6.59.3 Member Function Documentation

#### 6.59.3.1 void LuaInterface::init ( Game \* game ) [static]

Sets the lua\_this pointer and registers all C++ API functions to Lua.

Parameters

<i>Pointer</i>	to the game object that provides the game data to Lua.
----------------	--

Definition at line 44 of file LuaInterface.cpp.

#### 6.59.3.2 int LuaInterface::lua\_get\_avg\_fps ( lpp::Script::state L ) [static],[private]

Note

Function definitions below act as an interface between C++ and Lua, they all have to have the signature `int fname(lpp::Script::state)` and return the number of results pushed onto the Lua stack (Lua allows to return multiple results if needed). Important: These functions will have their arguments on the stack in REVERSED ORDER! (Because, you know, it's a stack...)

Definition at line 892 of file LuaInterface.cpp.

### 6.59.4 Member Data Documentation

#### 6.59.4.1 Game \* LuaInterface::lua\_this {} [static],[private]

Lua requires all functions registered in it to be static (because it doesn't know anything about C++).

Static member initialization, will be set in the init method.

Definition at line 34 of file LuaInterface.hpp.

The documentation for this class was generated from the following files:

- LuaInterface.hpp
- LuaInterface.cpp

## 6.60 ManaComponent Struct Reference

Allows an entity to cast spell by providing the mana resource.

```
#include <Components.hpp>
```

### Public Member Functions

- **ManaComponent** (tdt::uint max=0, tdt::uint regen=0)
- **ManaComponent** (const [ManaComponent](#) &)=default
- **ManaComponent** ([ManaComponent](#) &&)=default
- [ManaComponent](#) & **operator=** (const [ManaComponent](#) &)=default
- [ManaComponent](#) & **operator=** ([ManaComponent](#) &&)=default

### Public Attributes

- tdt::uint **curr\_mana**
- tdt::uint **max\_mana**
- tdt::uint **mana\_regen**

### Static Public Attributes

- static constexpr int **type** = 9

#### 6.60.1 Detailed Description

Allows an entity to cast spell by providing the mana resource.

Definition at line 251 of file Components.hpp.

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

- Components.hpp

## 6.61 ManaCrystalComponent Struct Reference

Allows an entity to increase the player's mana capacity and regeneration rate while it's alive.

```
#include <Components.hpp>
```

### Public Member Functions

- **ManaCrystalComponent** (tdt::uint cap=0, tdt::uint regen=0)
- **ManaCrystalComponent** (const [ManaCrystalComponent](#) &)=default
- **ManaCrystalComponent** ([ManaCrystalComponent](#) &&)=default
- [ManaCrystalComponent](#) & **operator=** (const [ManaCrystalComponent](#) &)=default
- [ManaCrystalComponent](#) & **operator=** ([ManaCrystalComponent](#) &&)=default

### Public Attributes

- tdt::uint **cap\_increase**
- tdt::uint **regen\_increase**

### Static Public Attributes

- static constexpr int **type** = 26

#### 6.61.1 Detailed Description

Allows an entity to increase the player's mana capacity and regeneration rate while it's alive.

Definition at line 626 of file Components.hpp.

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

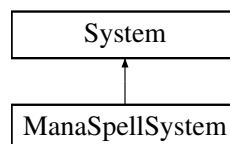
- Components.hpp

## 6.62 ManaSpellSystem Class Reference

Regenerates mana to the player and all entities that have mana.

```
#include <ManaSpellSystem.hpp>
```

Inheritance diagram for ManaSpellSystem:



### Public Member Functions

- [ManaSpellSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~ManaSpellSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Regenerates mana if necessary and performs entity spell casting if off cooldown.*
- void [set\\_regen\\_period](#) (tdt::real)  
*Sets the time period between mana regens.*
- tdt::real [get\\_regen\\_period](#) () const  
*Returns the time period between mana regens.*

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing entities that this system works with.*
- tdt::real [regen\\_timer\\_](#)  
*Allow for dynamic periods between mana regeneration updates.*
- tdt::real [regen\\_period\\_](#)

### 6.62.1 Detailed Description

Regenerates mana to the player and all entities that have mana.

Also takes care of entity spell casting.

Definition at line 11 of file ManaSpellSystem.hpp.

### 6.62.2 Constructor & Destructor Documentation

#### 6.62.2.1 ManaSpellSystem::ManaSpellSystem ( EntitySystem & ents )

Constructor.

Parameters

<i>Entity</i>	system containing entities this system works with.
---------------	--

Definition at line 6 of file ManaSpellSystem.cpp.

#### 6.62.2.2 ManaSpellSystem::~ManaSpellSystem ( ) [default]

Destructor.

### 6.62.3 Member Function Documentation

#### 6.62.3.1 tdt::real ManaSpellSystem::get\_regen\_period ( ) const

Returns the time period between mana regens.

Definition at line 45 of file ManaSpellSystem.cpp.

#### 6.62.3.2 void ManaSpellSystem::set\_regen\_period ( tdt::real val )

Sets the time period between mana regens.



## Parameters

<i>The</i>	new time period.
------------	------------------

Definition at line 40 of file ManaSpellSystem.cpp.

**6.62.3.3** void ManaSpellSystem::update ( tdt::real *delta* ) [override],[virtual]

Regenerates mana if necessary and performs entity spell casting if off cooldown.

## Parameters

<i>Time</i>	since last frame.
-------------	-------------------

Implements [System](#).

Definition at line 10 of file ManaSpellSystem.cpp.

**6.62.4 Member Data Documentation**

**6.62.4.1** EntitySystem& ManaSpellSystem::entities\_ [private]

Entity system containing entities that this system works with.

Definition at line 48 of file ManaSpellSystem.hpp.

**6.62.4.2** tdt::real ManaSpellSystem::regen\_timer\_ [private]

Allow for dynamic periods between mana regeneration updates.

Definition at line 54 of file ManaSpellSystem.hpp.

The documentation for this class was generated from the following files:

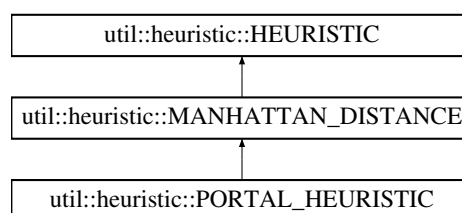
- systems/ManaSpellSystem.hpp
- systems/ManaSpellSystem.cpp

**6.63 util::heuristic::MANHATTAN\_DISTANCE Struct Reference**

Returns the manhattan distance between two nodes.

```
#include <PathfindingAlgorithms.hpp>
```

Inheritance diagram for util::heuristic::MANHATTAN\_DISTANCE:



## Public Member Functions

- **MANHATTAN\_DISTANCE** ([EntitySystem](#) &ents)
- tdt::real **get\_cost** (tdt::uint id1, tdt::uint id2) override

## Additional Inherited Members

### 6.63.1 Detailed Description

Returns the manhattan distance between two nodes.

(Well, actually, it's an octal distance :)

Definition at line 206 of file PathfindingAlgorithms.hpp.

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

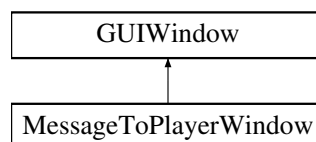
- tools/PathfindingAlgorithms.hpp

## 6.64 MessageToPlayerWindow Class Reference

A window that can show the player a text message with 1, 2 or 3 buttons (custom labels) that can call assigned functions.

```
#include <MessageToPlayerWindow.hpp>
```

Inheritance diagram for MessageToPlayerWindow:



## Public Member Functions

- [MessageToPlayerWindow](#) ()  
*Constructor.*
- [~MessageToPlayerWindow](#) ()=default  
*Destructor.*
- void [show](#) (const std::string &="NONE", const std::string &="NONE", const std::string &="NONE", const std::string &="NONE")  
*Shows a given text message to the player and assigns callbacks to the buttons.*
- void [set\\_but\\_label](#) (const std::string &, const std::string &)  
*Sets the label of a given button.*
- void [reset\\_but\\_labels](#) ()  
*Resets the button labels to their default values.*

## Protected Member Functions

- void `init_()`  
*Initializes this window.*

## Private Attributes

- `lpp::Script & script_`  
*Reference to the scripting engine for easier use when calling Lua callbacks.*
- `std::string ok_func_`  
*Labels of the buttons.*
- `std::string yes_func_`
- `std::string no_func_`

## Additional Inherited Members

### 6.64.1 Detailed Description

A window that can show the player a text message with 1, 2 or 3 buttons (custom labels) that can call assigned functions.

Button names: (used for setting labels) 1st on the left: NO 2nd on the left: YES 1st on the right: OK

Definition at line 18 of file `MessageToPlayerWindow.hpp`.

### 6.64.2 Constructor & Destructor Documentation

#### 6.64.2.1 `MessageToPlayerWindow::MessageToPlayerWindow ( )`

Constructor.

Definition at line 5 of file `MessageToPlayerWindow.cpp`.

#### 6.64.2.2 `MessageToPlayerWindow::~~MessageToPlayerWindow ( )` [default]

Destructor.

### 6.64.3 Member Function Documentation

#### 6.64.3.1 `void MessageToPlayerWindow::init_ ( )` [protected],[virtual]

Initializes this window.

Implements [GUIWindow](#).

Definition at line 53 of file `MessageToPlayerWindow.cpp`.

#### 6.64.3.2 `void MessageToPlayerWindow::reset_butt_labels ( )`

Resets the button labels to their default values.

("OK", "YES" and "NO")

Definition at line 46 of file `MessageToPlayerWindow.cpp`.

#### 6.64.3.3 `void MessageToPlayerWindow::set_butt_label ( const std::string & butt, const std::string & val )`

Sets the label of a given button.

## Parameters

<i>The</i>	name of the button (OK, YES, NO).
<i>The</i>	new label.

Definition at line 41 of file MessageToPlayerWindow.cpp.

**6.64.3.4** `void MessageToPlayerWindow::show ( const std::string & msg = "NONE", const std::string & ok = "NONE", const std::string & yes = "NONE", const std::string & no = "NONE" )`

Shows a given text message to the player and assigns callbacks to the buttons.

## Parameters

<i>The</i>	message.
<i>Callback</i>	for the OK button.
<i>Callback</i>	for the YES button.
<i>Callback</i>	for the NO button.

## Note

If a callback passes is "NONE", the assigned button will not be shown.  
The callbacks are strings of names of the Lua functions that ought to be called when the button is pressed.

Definition at line 10 of file MessageToPlayerWindow.cpp.

**6.64.4 Member Data Documentation**

**6.64.4.1** `std::string MessageToPlayerWindow::ok_func_ [private]`

Labels of the buttons.

Definition at line 74 of file MessageToPlayerWindow.hpp.

**6.64.4.2** `lpp::Script& MessageToPlayerWindow::script_ [private]`

Reference to the scripting engine for easier use when calling Lua callbacks.

Definition at line 69 of file MessageToPlayerWindow.hpp.

The documentation for this class was generated from the following files:

- gui/MessageToPlayerWindow.hpp
- gui/MessageToPlayerWindow.cpp

## 6.65 MineComponent Struct Reference

Dummy component that signals that an entity having it can be mined.

```
#include <Components.hpp>
```

### Static Public Attributes

- static constexpr int **type** = 25

### 6.65.1 Detailed Description

Dummy component that signals that an entity having it can be mined.

Definition at line 617 of file Components.hpp.

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

- Components.hpp

## 6.66 MovementComponent Struct Reference

Holds info related to movement, if an entity has this component it should also have a Physics component (containing the entity's position), otherwise the [MovementSystem](#) might not work correctly.

```
#include <Components.hpp>
```

### Public Member Functions

- **MovementComponent** (tdt::real speed=0.f)
- **MovementComponent** (const [MovementComponent](#) &)=default
- **MovementComponent** ([MovementComponent](#) &&)=default
- [MovementComponent](#) & **operator=** (const [MovementComponent](#) &)=default
- [MovementComponent](#) & **operator=** ([MovementComponent](#) &&)=default

### Public Attributes

- tdt::real **speed\_modifier**
- tdt::real **original\_speed**

### Static Public Attributes

- static constexpr int **type** = 4

### 6.66.1 Detailed Description

Holds info related to movement, if an entity has this component it should also have a Physics component (containing the entity's position), otherwise the [MovementSystem](#) might not work correctly.

Definition at line 132 of file Components.hpp.

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

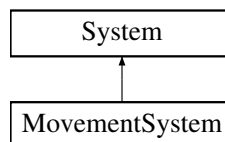
- Components.hpp

## 6.67 MovementSystem Class Reference

[System](#) handling movement related updates and containing movement & physics related methods.

```
#include <MovementSystem.hpp>
```

Inheritance diagram for MovementSystem:



### Public Member Functions

- [MovementSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~MovementSystem](#) ()  
*Destructor.*
- void [update](#) (Ogre::Real)  
*Updates the movement system.*
- bool [can\\_move\\_to](#) (std::size\_t, Ogre::Vector3)  
*Returns true if a given entity can move to a given point in space, false otherwise.*
- bool [checked\\_move](#) (std::size\_t, Ogre::Vector3)  
*Sets a given entity to move in a given direction, returns true if such movement is possible and false otherwise.*
- bool [move](#) (std::size\_t, Ogre::Vector3)  
*Sets a given entity to move in a given direction, returns true if such movement is possible and false otherwise.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- Ogre::Real [last\\_delta\\_](#)  
*Time between the last frame and this frame, used so that lua calls to the move functions still use the time of this frame.*

### 6.67.1 Detailed Description

[System](#) handling movement related updates and containing movement & physics related methods.

Definition at line 10 of file MovementSystem.hpp.

### 6.67.2 Constructor & Destructor Documentation

#### 6.67.2.1 MovementSystem::MovementSystem ( EntitySystem & *ents* )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 7 of file MovementSystem.cpp.

#### 6.67.2.2 MovementSystem::~~MovementSystem ( ) [inline]

Destructor.

Definition at line 22 of file MovementSystem.hpp.

### 6.67.3 Member Function Documentation

#### 6.67.3.1 bool MovementSystem::can\_move\_to ( std::size\_t *id*, Ogre::Vector3 *pos* )

Returns true if a given entity can move to a given point in space, false otherwise.

Parameters

<i>ID</i>	of the entity.
<i>Target</i>	coordinate.

Definition at line 49 of file MovementSystem.cpp.

#### 6.67.3.2 bool MovementSystem::checked\_move ( std::size\_t *id*, Ogre::Vector3 *dir\_vector* )

Sets a given entity to move in a given direction, returns true if such movement is possible and false otherwise.

The move will then be applied in the update method.

Parameters

<i>ID</i>	of the entity.
<i>Directional</i>	vector.

**Note**

Every vector passed to this method should be normalised and the length of the move will be increased by the entity's speed modifier. This is not enforced though.  
Checks for collisions.

Definition at line 85 of file MovementSystem.cpp.

### 6.67.3.3 `bool MovementSystem::move ( std::size_t id, Ogre::Vector3 dir_vector )`

Sets a given entity to move in a given direction, returns true if such movement is possible and false otherwise.

The move will then be applied in the update method.

**Parameters**

<i>ID</i>	of the entity.
<i>Directional</i>	vector.

**Note**

Every vector passed to this method should be normalised and the length of the move will be increased by the entity's speed modifier. This is not enforced though.  
Does not check for collisions.

Definition at line 110 of file MovementSystem.cpp.

### 6.67.3.4 `void MovementSystem::update ( Ogre::Real delta ) [virtual]`

Updates the movement system.

**Parameters**

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 11 of file MovementSystem.cpp.

## 6.67.4 Member Data Documentation

### 6.67.4.1 `EntitySystem& MovementSystem::entities_ [private]`

Reference to the game's entity system.

Definition at line 63 of file MovementSystem.hpp.



#### 6.67.4.2 Ogre::Real MovementSystem::last\_delta\_ [private]

Time between the last frame and this frame, used so that lua calls to the move functions still use the time of this frame.

Definition at line 69 of file MovementSystem.hpp.

The documentation for this class was generated from the following files:

- systems/MovementSystem.hpp
- systems/MovementSystem.cpp

## 6.68 NameComponent Struct Reference

Name of the entity shown in the entity viewer.

```
#include <Components.hpp>
```

### Public Member Functions

- **NameComponent** (std::string &&n="ERROR")
- **NameComponent** (const [NameComponent](#) &)=default
- **NameComponent** ([NameComponent](#) &&)=default
- [NameComponent](#) & **operator=** (const [NameComponent](#) &)=default
- [NameComponent](#) & **operator=** ([NameComponent](#) &&)=default

### Public Attributes

- std::string **name**

### Static Public Attributes

- static constexpr int **type** = 34

#### 6.68.1 Detailed Description

Name of the entity shown in the entity viewer.

Definition at line 803 of file Components.hpp.

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

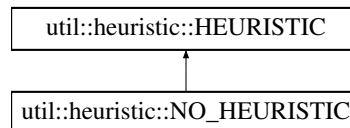
- Components.hpp

## 6.69 util::heuristic::NO\_HEURISTIC Struct Reference

Represents no heuristic by returning 0 all the time.

```
#include <PathfindingAlgorithms.hpp>
```

Inheritance diagram for util::heuristic::NO\_HEURISTIC:



### Public Member Functions

- **NO\_HEURISTIC** ([EntitySystem](#) &ents)
- tdt::real **get\_cost** (tdt::uint id1, tdt::uint id2) override

### Additional Inherited Members

#### 6.69.1 Detailed Description

Represents no heuristic by returning 0 all the time.

Definition at line 223 of file PathfindingAlgorithms.hpp.

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

- tools/PathfindingAlgorithms.hpp

## 6.70 NotificationComponent Struct Reference

Allows to keep track about notification cooldown, so that an entity doesn't spam the player with messages on reoccurring events in a short time period.

```
#include <Components.hpp>
```

### Public Member Functions

- **NotificationComponent** (tdt::real cd=0.f)
- **NotificationComponent** (const [NotificationComponent](#) &)=default
- **NotificationComponent** ([NotificationComponent](#) &&)=default
- [NotificationComponent](#) & **operator=** (const [NotificationComponent](#) &)=default
- [NotificationComponent](#) & **operator=** ([NotificationComponent](#) &&)=default

### Public Attributes

- tdt::real **curr\_time**
- tdt::real **cooldown**

### Static Public Attributes

- static constexpr int **type** = 31

#### 6.70.1 Detailed Description

Allows to keep track about notification cooldown, so that an entity doesn't spam the player with messages on reoccurring events in a short time period.

Definition at line 739 of file Components.hpp.

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

- Components.hpp

## 6.71 OnHitComponent Struct Reference

Contains the blueprint table which gets called when an entity that has this component gets hit.

```
#include <Components.hpp>
```

### Public Member Functions

- **OnHitComponent** (std::string &&b="ERROR", tdt::real cd=0.f)
- **OnHitComponent** (const [OnHitComponent](#) &)=default
- **OnHitComponent** ([OnHitComponent](#) &&)=default
- [OnHitComponent](#) & **operator=** (const [OnHitComponent](#) &)=default
- [OnHitComponent](#) & **operator=** ([OnHitComponent](#) &&)=default

### Public Attributes

- std::string **blueprint**
- tdt::real **curr\_time**
- tdt::real **cooldown**

### Static Public Attributes

- static constexpr int **type** = 27

### 6.71.1 Detailed Description

Contains the blueprint table which gets called when an entity that has this component gets hit.

Definition at line 647 of file Components.hpp.

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

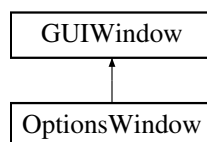
- Components.hpp

## 6.72 OptionsWindow Class Reference

Options menu window that lets the player to change the resolution, window mode and keybinding.

```
#include <OptionsWindow.hpp>
```

Inheritance diagram for OptionsWindow:



### Public Member Functions

- [OptionsWindow](#) ()  
*Constructor.*
- [~OptionsWindow](#) ()=default  
*Destructor.*
- void [add\\_start\\_parameters](#) (Ogre::RenderWindow \*, Ogre::Viewport \*, CEGUI::OgreRenderer \*)  
*Saves pointers to the window, view and renderer and sets starting values for the resolution, window mode etc.*
- bool [key\\_pressed](#) (CEGUI::Key::Scan)  
*Registers a key press for keybinding.*
- void [set\\_key\\_bind](#) (KEY\_BIND\_ACTION::VAL, CEGUI::Key::Scan)  
*Binds a given action to a given key.*

### Protected Member Functions

- void [init\\_](#) () override  
*Initializes the CEGUI window.*

### Private Types

- typedef void(\* [ActionFuncPtr](#)) ()

## Private Member Functions

- void [apply\\_\(\)](#)  
*Applies the graphical changes and saves key bindings into a script that allows persistent key binds.*
- void [update\\_labels\\_\(\)](#)  
*Updates the button texts and labels to reflect the new settings.*
- void [update\\_fonts\\_\(\)](#)
- void [update\\_font\\_of\\_window\\_\(\)](#) (CEGUI::Window \*, const std::string &)
- const std::string & [get\\_key\\_bind\\_name\\_\(\)](#) (KEY\_BIND\_ACTION::VAL)  
*Returns the name of the key a given action is bound to.*

## Private Attributes

- Ogre::RenderWindow \* [render\\_window\\_](#)  
*Window the game is rendered to.*
- Ogre::Viewport \* [view\\_](#)  
*Main viewport of the render window.*
- CEGUI::OgreRenderer \* [renderer\\_](#)  
*Renderer used by CEGUI (used to sync resolution between game and UI).*
- tdt::uint [width\\_](#)  
*Dimensions of the display resolution.*
- tdt::uint [height\\_](#)
- bool [fullscreen\\_](#)  
*True for fullscreen, false for windowed mode.*
- std::map< CEGUI::Key::Scan, std::string > [key\\_names\\_](#)  
*This map connects key codes with their names (used for key bind buttons).*
- std::array< ActionFuncPtr, KEY\_BIND\_ACTION::COUNT > [actions\\_](#)  
*Action functions assigned to actions.*
- std::array< CEGUI::Key::Scan, KEY\_BIND\_ACTION::COUNT > [key\\_binds\\_](#)  
*Keys bound to actions.*
- std::map< std::string, std::tuple< tdt::uint, tdt::uint > > [resolutions\\_](#)  
*Resolution strings and their assigned dimensions.*
- KEY\_BIND\_ACTION::VAL [currently\\_binded\\_action\\_](#)  
*Saves the currently binded action so that the next key press can be used.*

## Additional Inherited Members

### 6.72.1 Detailed Description

Options menu window that lets the player to change the resolution, window mode and keybinding.

Definition at line 18 of file OptionsWindow.hpp.

### 6.72.2 Constructor & Destructor Documentation

#### 6.72.2.1 OptionsWindow::OptionsWindow ( )

Constructor.

Definition at line 8 of file OptionsWindow.cpp.

#### 6.72.2.2 OptionsWindow::~~OptionsWindow ( ) [default]

Destructor.

### 6.72.3 Member Function Documentation

#### 6.72.3.1 void OptionsWindow::add\_start\_parameters ( Ogre::RenderWindow \* *window*, Ogre::Viewport \* *view*, CEGUI::OgreRenderer \* *renderer* )

Saves pointers to the window, view and renderer and sets starting values for the resolution, window mode etc.

##### Parameters

<i>Window</i>	that the game is rendered to.
<i>Viewport</i>	of the window.
<i>Renderer</i>	used by CEGUI.

Definition at line 13 of file OptionsWindow.cpp.

#### 6.72.3.2 void OptionsWindow::apply\_ ( ) [private]

Applies the graphical changes and saves key bindings into a script that allows persistent key binds.

This will make sure that key bindings are persistent.

Definition at line 307 of file OptionsWindow.cpp.

#### 6.72.3.3 const std::string & OptionsWindow::get\_key\_bind\_name\_ ( KEY\_BIND\_ACTION::VAL *action* ) [private]

Returns the name of the key a given action is bound to.

##### Parameters

<i>The</i>	action.
------------	---------

Definition at line 425 of file OptionsWindow.cpp.

#### 6.72.3.4 void OptionsWindow::init\_ ( ) [override],[protected],[virtual]

Initializes the CEGUI window.

Implements [GUIWindow](#).

Definition at line 64 of file OptionsWindow.cpp.

**6.72.3.5** `bool OptionsWindow::key_pressed ( CEGUI::Key::Scan key )`

Registers a key press for keybinding.

Returns true if the click was consumed, false otherwise.

**Parameters**

<i>Key</i>	pressed.
------------	----------

Definition at line 26 of file OptionsWindow.cpp.

**6.72.3.6** `void OptionsWindow::set_key_bind ( KEY_BIND_ACTION::VAL action, CEGUI::Key::Scan key )`

Binds a given action to a given key.

**Parameters**

<i>Action</i>	to be bound.
<i>Key</i>	to be bound.

Definition at line 58 of file OptionsWindow.cpp.

**6.72.3.7** `void OptionsWindow::update_labels_ ( )` `[private]`

Updates the button texts and labels to reflect the new settings.

Definition at line 348 of file OptionsWindow.cpp.

**6.72.4 Member Data Documentation****6.72.4.1** `std::array<ActionFuncPtr, KEY_BIND_ACTION::COUNT> OptionsWindow::actions_` `[private]`

Action functions assigned to actions.

Definition at line 124 of file OptionsWindow.hpp.

**6.72.4.2** `KEY_BIND_ACTION::VAL OptionsWindow::currently_binded_action_` `[private]`

Saves the currently binded action so that the next key press can be used.

Definition at line 140 of file OptionsWindow.hpp.

**6.72.4.3** `bool OptionsWindow::fullscreen_` `[private]`

True for fullscreen, false for windowed mode.

Definition at line 114 of file OptionsWindow.hpp.

**6.72.4.4** `std::array<CEGUI::Key::Scan, KEY_BIND_ACTION::COUNT> OptionsWindow::key_binds_` `[private]`

Keys bound to actions.

Definition at line 129 of file OptionsWindow.hpp.

**6.72.4.5** `std::map<CEGUI::Key::Scan, std::string> OptionsWindow::key_names_` `[private]`

This map connects key codes with their names (used for key bind buttons).

Definition at line 119 of file OptionsWindow.hpp.

**6.72.4.6** `Ogre::RenderWindow* OptionsWindow::render_window_` `[private]`

Window the game is rendered to.

Definition at line 94 of file OptionsWindow.hpp.

**6.72.4.7** `CEGUI::OgreRenderer* OptionsWindow::renderer_` `[private]`

Renderer used by CEGUI (used to sync resolution between game and UI).

Definition at line 104 of file OptionsWindow.hpp.

**6.72.4.8** `std::map<std::string, std::tuple<tdt::uint, tdt::uint> > OptionsWindow::resolutions_` `[private]`

Resolution strings and their assigned dimensions.

Definition at line 134 of file OptionsWindow.hpp.

**6.72.4.9** `Ogre::Viewport* OptionsWindow::view_` `[private]`

Main viewport of the render window.

Definition at line 99 of file OptionsWindow.hpp.

**6.72.4.10** `tdt::uint OptionsWindow::width_` `[private]`

Dimensions of the display resolution.

Definition at line 109 of file OptionsWindow.hpp.

The documentation for this class was generated from the following files:

- gui/OptionsWindow.hpp
- gui/OptionsWindow.cpp



## 6.73 PathfindingComponent Struct Reference

Holds data related to the entity's current path.

```
#include <Components.hpp>
```

### Public Member Functions

- **PathfindingComponent** (std::string &&b="ERROR", tdt::uint tar=0, tdt::uint last=0)
- **PathfindingComponent** (const [PathfindingComponent](#) &)=default
- **PathfindingComponent** ([PathfindingComponent](#) &&)=default
- [PathfindingComponent](#) & **operator=** (const [PathfindingComponent](#) &)=default
- [PathfindingComponent](#) & **operator=** ([PathfindingComponent](#) &&)=default

### Public Attributes

- tdt::uint **target\_id**
- tdt::uint **last\_id**
- std::deque< tdt::uint > **path\_queue**
- std::string **blueprint**

### Static Public Attributes

- static constexpr int **type** = 14

### 6.73.1 Detailed Description

Holds data related to the entity's current path.

Definition at line 367 of file Components.hpp.

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

- Components.hpp

## 6.74 PhysicsComponent Struct Reference

Components.

```
#include <Components.hpp>
```

### Public Member Functions

- **PhysicsComponent** (bool s=false, Ogre::Vector3 pos=Ogre::Vector3{0, 0, 0}, tdt::real hh=0.f)
- **PhysicsComponent** (const [PhysicsComponent](#) &)=default
- **PhysicsComponent** ([PhysicsComponent](#) &&)=default
- [PhysicsComponent](#) & **operator=** (const [PhysicsComponent](#) &)=default
- [PhysicsComponent](#) & **operator=** ([PhysicsComponent](#) &&)=default

## Public Attributes

- bool **solid**
- Ogre::Vector3 **position**
- tdt::real **half\_height**

## Static Public Attributes

- static constexpr int **type** = 0

### 6.74.1 Detailed Description

Components.

#### Note

To be able to manually create components without blueprints, all components must have either default constructors or constructors with default values for all parameters, to be able to back components up, they need to provide a copy constructor.

As constructors are called in the [EntitySystem::load\\_component](#) function, strings will be read from Lua and immediately discarded. Hence it was decided to make the string parameters accepted as rvalues for faster construction. Holds info related to physical interaction of an entity with the rest of the game world.

Definition at line 34 of file Components.hpp.

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

- Components.hpp

## 6.75 Player Class Reference

Auxiliary class representing the player's resources, since the nature of the game does not allow creating player as an entity (only one player can exist at a time) and it allows for easy [GUI](#) modifications when the amount of these resources changes (had player been an entity, ever gold/mana addition would require a check which would be true only in minimal number of cases).

```
#include <Player.hpp>
```

## Public Member Functions

- [~Player](#) ()  
*Destructor.*
- void [add\\_gold](#) (tdt::uint)  
*Adds gold to the player's gold stash.*
- bool [sub\\_gold](#) (tdt::uint)  
*Removes gold from the player's gold stash if possible, returns true if the player has enough, false otherwise.*
- void [add\\_mana](#) (tdt::uint)  
*Adds mana to the player's mana pool.*
- bool [sub\\_mana](#) (tdt::uint)  
*Removes mana from the player's mana pool if possible, returns true if the player has enough, false otherwise.*
- void [add\\_max\\_mana](#) (tdt::uint)  
*Breif: Increases the mana capacity of the player by a given amount.*
- bool [sub\\_max\\_mana](#) (tdt::uint)  
*Decreases the mana capacity of the player by a given amount if possible.*
- void [add\\_mana\\_regen](#) (tdt::uint)  
*Increases the player's mana regeneration by a given amount.*
- bool [sub\\_mana\\_regen](#) (tdt::uint)  
*Decreases the player's mana regeneration by a given amount if possible.*
- void [add\\_max\\_unit](#) (tdt::uint)  
*Adds max units to the player's unit amount.*
- bool [sub\\_max\\_unit](#) (tdt::uint)  
*Removes max units from the player's unit amount if possible, returns true if the player has enough, false otherwise.*
- void [add\\_curr\\_unit](#) (tdt::uint)  
*Adds current units to the player's unit amount.*
- bool [sub\\_curr\\_unit](#) (tdt::uint)  
*Removes current units from the player's unit amount if possible, returns true if the player has enough, false otherwise.*
- tdt::uint [get\\_gold](#) () const  
*Returns the amount of gold the player currently has.*
- tdt::uint [get\\_mana](#) () const  
*Returns the amount of mana the player currently has.*
- tdt::uint [get\\_max\\_mana](#) () const  
*Returns the mana capacity of the player.*
- tdt::uint [get\\_mana\\_regen](#) () const  
*Returns the value of the player's mana regeneration.*
- void [reset](#) ()  
*Sets all of the player's stats to their default values.*
- void [nulify\\_all\\_stats](#) ()  
*Sets all of the player's stats to zero (used for loading).*
- void [init](#) (EntitySystem \*)  
*Initializes the player class.*
- void [set\\_initial\\_unlocks](#) (const std::vector< std::string > &, const std::vector< std::string > &)  
*Sets the spells and buildings that are unlocked from the start.*
- const std::vector< std::string > & [get\\_initial\\_spells](#) () const  
*Returns a vector of all spells that are unlocked at the start of a new game.*
- const std::vector< std::string > & [get\\_initial\\_buildings](#) () const  
*Returns a vector of all buildings that are unlocked at the start of a new game.*

## Static Public Member Functions

- static [Player](#) & [instance](#) ()  
*Returns a reference to the singleton instance.*

## Private Member Functions

- [Player](#) ()  
*Constructor.*

## Private Attributes

- `tdt::uint` [gold\\_](#)  
*Amount of total gold the player currently has to spend.*
- `tdt::uint` [mana\\_](#)  
*Amount of mana the player currently has to spend.*
- `tdt::uint` [max\\_mana\\_](#)  
*Max amount of mana the player can have.*
- `tdt::uint` [mana\\_regen\\_](#)  
*Amount of mana that is added to the player's mana pool on every regen tick.*
- `tdt::uint` [units\\_curr\\_](#)  
*Amount of currently alive units.*
- `tdt::uint` [units\\_max\\_](#)  
*Amount of all units (even those that are respawning).*
- `const tdt::uint` [uint\\_max\\_](#)  
*Helper value for overflow checking, contains tdt::uint max value.*
- [EntitySystem](#) \* [entities\\_](#)  
*Used to subtract gold from gold vault so that player gold and vault gold is synchronized.*
- `std::vector< std::string >` [initial\\_spell\\_unlocks\\_](#)  
*Holds the names of the spells that are available to the player from the beginning.*
- `std::vector< std::string >` [initial\\_building\\_unlocks\\_](#)  
*Holds the names of the buildings that are available to the player from the beginning.*

### 6.75.1 Detailed Description

Auxiliary class representing the player's resources, since the nature of the game does not allow creating player as an entity (only one player can exist at a time) and it allows for easy [GUI](#) modifications when the amount of these resources changes (had player been an entity, ever gold/mana addition would require a check which would be true only in minimal number of cases).

Definition at line 15 of file `Player.hpp`.

### 6.75.2 Constructor & Destructor Documentation

#### 6.75.2.1 `Player::~~Player ( )` [`inline`]

Destructor.

Definition at line 21 of file `Player.hpp`.

### 6.75.2.2 Player::Player ( ) [private]

Constructor.

Definition at line 30 of file Player.cpp.

## 6.75.3 Member Function Documentation

### 6.75.3.1 void Player::add\_curr\_unit ( tdt::uint val )

Adds current units to the player's unit amount.

#### Parameters

<i>Amount</i>	to add.
---------------	---------

Definition at line 174 of file Player.cpp.

### 6.75.3.2 void Player::add\_gold ( tdt::uint val )

Adds gold to the player's gold stash.

#### Parameters

<i>Amount</i>	to add.
---------------	---------

Definition at line 35 of file Player.cpp.

### 6.75.3.3 void Player::add\_mana ( tdt::uint val )

Adds mana to the player's mana pool.

#### Parameters

<i>Amount</i>	to add.
---------------	---------

Definition at line 71 of file Player.cpp.

### 6.75.3.4 void Player::add\_mana\_regen ( tdt::uint val )

Increases the player's mana regeneration by a given amount.

#### Parameters

<i>Amount</i>	to increase by.
---------------	-----------------

Definition at line 126 of file Player.cpp.

#### 6.75.3.5 void Player::add\_max\_mana ( tdt::uint val )

Breif: Increases the mana capacity of the player by a given amount.

##### Parameters

<i>Amount</i>	to add.
---------------	---------

Definition at line 101 of file Player.cpp.

#### 6.75.3.6 void Player::add\_max\_unit ( tdt::uint val )

Adds max units to the player's unit amount.

##### Parameters

<i>Amount</i>	to add.
---------------	---------

Definition at line 151 of file Player.cpp.

#### 6.75.3.7 tdt::uint Player::get\_gold ( ) const

Returns the amount of gold the player currently has.

Definition at line 197 of file Player.cpp.

#### 6.75.3.8 const std::vector< std::string > & Player::get\_initial\_buildings ( ) const

Returns a vector of all buildings that are unlocked at the start of a new game.

Definition at line 25 of file Player.cpp.

#### 6.75.3.9 const std::vector< std::string > & Player::get\_initial\_spells ( ) const

Returns a vector of all spells that are unlocked at the start of a new game.

Definition at line 20 of file Player.cpp.

#### 6.75.3.10 tdt::uint Player::get\_mana ( ) const

Returns the amount of mana the player currently has.

Definition at line 202 of file Player.cpp.

**6.75.3.11** `tdt::uint Player::get_mana_regen ( ) const`

Returns the value of the player's mana regeneration.

Definition at line 212 of file Player.cpp.

**6.75.3.12** `tdt::uint Player::get_max_mana ( ) const`

Returns the mana capacity of the player.

Definition at line 207 of file Player.cpp.

**6.75.3.13** `void Player::init ( EntitySystem * ents )`

Initializes the player class.

Parameters

<a href="#">EntitySystem</a>	used to get gold vaults to subtract gold from.
------------------------------	--

Definition at line 8 of file Player.cpp.

**6.75.3.14** `static Player& Player::instance ( ) [inline],[static]`

Returns a reference to the singleton instance.

Definition at line 140 of file Player.hpp.

**6.75.3.15** `void Player::nulify_all_stats ( )`

Sets all of the player's stats to zero (used for loading).

Definition at line 229 of file Player.cpp.

**6.75.3.16** `void Player::reset ( )`

Sets all of the player's stats to their default values.

Definition at line 217 of file Player.cpp.

**6.75.3.17** `void Player::set_initial_unlocks ( const std::vector< std::string > & spells, const std::vector< std::string > & buildings )`

Sets the spells and buildings that are unlocked from the start.

## Parameters

<i>Spell</i>	unlocks.
<i>Building</i>	unlocks.

Definition at line 14 of file Player.cpp.

**6.75.3.18** `bool Player::sub_curr_unit ( tdt::uint val )`

Removes current units from the player's unit amount if possible, returns true if the player has enough, false otherwise.

## Parameters

<i>Amount</i>	to remove.
---------------	------------

Definition at line 184 of file Player.cpp.

**6.75.3.19** `bool Player::sub_gold ( tdt::uint val )`

Removes gold from the player's gold stash if possible, returns true if the player has enough, false otherwise.

## Parameters

<i>Amount</i>	to remove.
---------------	------------

Definition at line 44 of file Player.cpp.

**6.75.3.20** `bool Player::sub_mana ( tdt::uint val )`

Removes mana from the player's mana pool if possible, returns true if the player has enough, false otherwise.

## Parameters

<i>Amount</i>	to remove.
---------------	------------

Definition at line 87 of file Player.cpp.

**6.75.3.21** `bool Player::sub_mana_regen ( tdt::uint val )`

Decreases the player's mana regeneration by a given amount if possible.

Returns true if the player has enough, false otherwise.



## Parameters

<i>Amount</i>	to decrease by.
---------------	-----------------

Definition at line 137 of file Player.cpp.

#### 6.75.3.22 bool Player::sub\_max\_mana ( tdt::uint val )

Decreases the mana capacity of the player by a given amount if possible.

Returns true if the player has enough, false otherwise.

## Parameters

<i>Amount</i>	to decrease by.
---------------	-----------------

Definition at line 112 of file Player.cpp.

#### 6.75.3.23 bool Player::sub\_max\_unit ( tdt::uint val )

Removes max units from the player's unit amount if possible, returns true if the player has enough, false otherwise.

## Parameters

<i>Amount</i>	to remove.
---------------	------------

Definition at line 161 of file Player.cpp.

### 6.75.4 Member Data Documentation

#### 6.75.4.1 EntitySystem\* Player::entities\_ [private]

Used to subtract gold from gold vault so that player gold and vault gold is synchronized.

Definition at line 219 of file Player.hpp.

#### 6.75.4.2 tdt::uint Player::gold\_ [private]

Amount of total gold the player currently has to spend.

## Note

Only gold stored in vaults is counted, not that on units.

Definition at line 182 of file Player.hpp.

#### 6.75.4.3 `std::vector<std::string> Player::initial_building_unlocks_` [private]

Holds the names of the buildings that are available to the player from the beginning.

(That is, they are unlocked in the Lua script they are defined.)

Definition at line 233 of file Player.hpp.

#### 6.75.4.4 `std::vector<std::string> Player::initial_spell_unlocks_` [private]

Holds the names of the spells that are available to the player from the beginning.

(That is, they are unlocked in the Lua script they are defined.)

Definition at line 226 of file Player.hpp.

#### 6.75.4.5 `tdt::uint Player::mana_` [private]

Amount of mana the player currently has to spend.

Definition at line 187 of file Player.hpp.

#### 6.75.4.6 `tdt::uint Player::mana_regen_` [private]

Amount of mana that is added to the player's mana pool on every regen tick.

Definition at line 198 of file Player.hpp.

#### 6.75.4.7 `tdt::uint Player::max_mana_` [private]

Max amount of mana the player can have.

Definition at line 192 of file Player.hpp.

#### 6.75.4.8 `const tdt::uint Player::uint_max_` [private]

Helper value for overflow checking, contains tdt::uint max value.

Definition at line 213 of file Player.hpp.

#### 6.75.4.9 `tdt::uint Player::units_curr_` [private]

Amount of currently alive units.

Definition at line 203 of file Player.hpp.

## 6.75.4.10 tdt::uint Player::units\_max\_ [private]

Amount of all units (even those that are respawning).

Definition at line 208 of file Player.hpp.

The documentation for this class was generated from the following files:

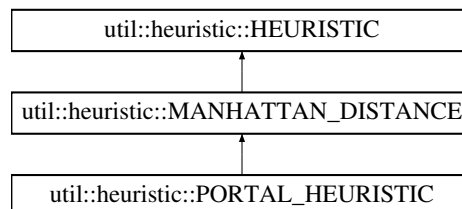
- tools/Player.hpp
- tools/Player.cpp

## 6.76 util::heuristic::PORTAL\_HEURISTIC Struct Reference

Variation of the Manhattan distance heuristic that takes portals into accounts.

```
#include <PathfindingAlgorithms.hpp>
```

Inheritance diagram for util::heuristic::PORTAL\_HEURISTIC:



### Public Member Functions

- **PORTAL\_HEURISTIC** ([EntitySystem](#) &ents)
- tdt::real **get\_cost** (tdt::uint id1, tdt::uint id2) override

### Private Member Functions

- std::tuple< tdt::uint, tdt::uint > [get\\_closest\\_portal](#) (tdt::uint id)  
*Returns the nodes that have the closest portal pair from a given entity on them.*

### Additional Inherited Members

#### 6.76.1 Detailed Description

Variation of the Manhattan distance heuristic that takes portals into accounts.

##### Note

This heuristic won't help with complex chains of portals. For that, the BEST\_PATH path type would be needed to check every single portal combination. (But for basic portal usage, this heuristic works fine.)

Definition at line 267 of file PathfindingAlgorithms.hpp.

#### 6.76.2 Member Function Documentation

6.76.2.1 `std::tuple<tdt::uint, tdt::uint> util::heuristic::PORTAL_HEURISTIC::get_closest_portal ( tdt::uint id ) [inline], [private]`

Returns the nodes that have the closest portal pair from a given entity on them.

#### Parameters

<i>ID</i>	of the entity.
-----------	----------------

Definition at line 298 of file PathfindingAlgorithms.hpp.

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

- tools/PathfindingAlgorithms.hpp

## 6.77 PortalComponent Struct Reference

Dummy component that signals that an entity having it is a portal - which is used in pathfinding.

```
#include <Components.hpp>
```

#### Static Public Attributes

- static constexpr int **type** = 39

### 6.77.1 Detailed Description

Dummy component that signals that an entity having it is a portal - which is used in pathfinding.

Definition at line 899 of file Components.hpp.

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

- Components.hpp

## 6.78 PriceComponent Struct Reference

Represents either gold or mana cost of an entity.

```
#include <Components.hpp>
```

#### Public Member Functions

- **PriceComponent** (tdt::uint p=0)
- **PriceComponent** (const [PriceComponent](#) &)=default
- **PriceComponent** ([PriceComponent](#) &&)=default
- [PriceComponent](#) & **operator=** (const [PriceComponent](#) &)=default
- [PriceComponent](#) & **operator=** ([PriceComponent](#) &&)=default

## Public Attributes

- tdt::uint **price**

## Static Public Attributes

- static constexpr int **type** = 23

### 6.78.1 Detailed Description

Represents either gold or mana cost of an entity.

Definition at line 568 of file Components.hpp.

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

- Components.hpp

## 6.79 ProductComponent Struct Reference

References the producer of the entity that has this component.

```
#include <Components.hpp>
```

## Public Member Functions

- **ProductComponent** (tdt::uint prod\_id=Component::NO\_ENTITY)
- **ProductComponent** (const [ProductComponent](#) &)=default
- **ProductComponent** ([ProductComponent](#) &&)=default
- [ProductComponent](#) & **operator=** (const [ProductComponent](#) &)=default
- [ProductComponent](#) & **operator=** ([ProductComponent](#) &&)=default

## Public Attributes

- tdt::uint **producer**

## Static Public Attributes

- static constexpr int **type** = 13

### 6.79.1 Detailed Description

References the producer of the entity that has this component.

(Producer == building/tile that spawned it.)

Definition at line 348 of file Components.hpp.

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

- Components.hpp

## 6.80 ProductionComponent Struct Reference

Allows scheduled production of new entities (spawners) of a given type up to a maximum amount.

```
#include <Components.hpp>
```

### Public Member Functions

- **ProductionComponent** (std::string &&b="ERROR", tdt::uint l=1, tdt::real cd=0.f)
- **ProductionComponent** (const [ProductionComponent](#) &)=default
- **ProductionComponent** ([ProductionComponent](#) &&)=default
- [ProductionComponent](#) & **operator=** (const [ProductionComponent](#) &)=default
- [ProductionComponent](#) & **operator=** ([ProductionComponent](#) &&)=default

### Public Attributes

- std::string **product\_blueprint**
- tdt::uint **curr\_produced**
- tdt::uint **max\_produced**
- tdt::real **cooldown**
- tdt::real **curr\_cd**

### Static Public Attributes

- static constexpr int **type** = 11

### 6.80.1 Detailed Description

Allows scheduled production of new entities (spawners) of a given type up to a maximum amount.

Definition at line 294 of file Components.hpp.

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

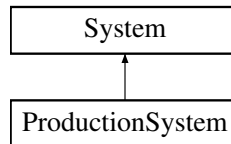
- Components.hpp

## 6.81 ProductionSystem Class Reference

[System](#) taking care of entities spawned by buildings and the spawn counts allowing for a constant amount of entities (related to the number of buildings spawning entities of that blueprint table).

```
#include <ProductionSystem.hpp>
```

Inheritance diagram for ProductionSystem:



### Public Member Functions

- [ProductionSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~ProductionSystem](#) ()  
*Destructor.*
- void [update](#) (tdt::real) override  
*Iterates over all buildings and when possible, spawns new entities.*
- void [spawn\\_entity](#) (tdt::uint, const std::string &)  
*Spawns a single entity created by a building.*
- void [set\\_time\\_multiplier](#) (tdt::real)  
*Sets the time value by which the frame times are multiplied when added to the production timers.*
- tdt::real [get\\_time\\_multiplier](#) ()  
*Returns the time value by which the frame times are multiplied when added to the production timers.*

### Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system (component retrieval).*
- [Grid](#) & [grid\\_](#)  
*Reference to the game's pathfinding grid (spawn positioning).*
- tdt::real [time\\_multiplier\\_](#)  
*Allows to speed up/slow down the production of all buildings.*

#### 6.81.1 Detailed Description

[System](#) taking care of entities spawned by buildings and the spawn counts allowing for a constant amount of entities (related to the number of buildings spawning entities of that blueprint table).

Definition at line 14 of file ProductionSystem.hpp.

#### 6.81.2 Constructor & Destructor Documentation

##### 6.81.2.1 ProductionSystem::ProductionSystem ( EntitySystem & ents )

Constructor.

## Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 7 of file ProductionSystem.cpp.

#### 6.81.2.2 `ProductionSystem::~~ProductionSystem ( )` `[inline]`

Destructor.

Definition at line 26 of file ProductionSystem.hpp.

### 6.81.3 Member Function Documentation

#### 6.81.3.1 `tdt::real ProductionSystem::get_time_multiplier ( )`

Returns the time value by which the frame times are multiplied when added to the production timers.

Definition at line 125 of file ProductionSystem.cpp.

#### 6.81.3.2 `void ProductionSystem::set_time_multiplier ( tdt::real val )`

Sets the time value by which the frame times are multiplied when added to the production timers.

## Parameters

<i>The</i>	new time multiplier.
------------	----------------------

Definition at line 120 of file ProductionSystem.cpp.

#### 6.81.3.3 `void ProductionSystem::spawn_entity ( tdt::uint producer, const std::string & blueprint )`

Spawns a single entity created by a building.

## Parameters

<i>ID</i>	of the building.
<i>Name</i>	of the blueprint table of the spawned entity.

This checks all edges of the building to find a free spot for the entity to spawn on.

Definition at line 29 of file ProductionSystem.cpp.



6.81.3.4 void ProductionSystem::update ( tdt::real *delta* ) [override],[virtual]

Iterates over all buildings and when possible, spawns new entities.

## Parameters

<i>Time</i>	since last frame.
-------------	-------------------

Implements [System](#).

Definition at line 11 of file ProductionSystem.cpp.

## 6.81.4 Member Data Documentation

### 6.81.4.1 EntitySystem& ProductionSystem::entities\_ [private]

Reference to the game's entity system (component retrieval).

Definition at line 58 of file ProductionSystem.hpp.

### 6.81.4.2 Grid& ProductionSystem::grid\_ [private]

Reference to the game's pathfinding grid (spawn positioning).

Definition at line 63 of file ProductionSystem.hpp.

### 6.81.4.3 tdt::real ProductionSystem::time\_multiplier\_ [private]

Allows to speed up/slow down the production of all buildings.

Definition at line 68 of file ProductionSystem.hpp.

The documentation for this class was generated from the following files:

- systems/ProductionSystem.hpp
- systems/ProductionSystem.cpp

## 6.82 util::path\_type::RANDOM\_PATH< UPPER > Struct Template Reference

Finds a random path by returning true only when a random number in the range (0, UPPER) is equal to 0.

```
#include <PathfindingAlgorithms.hpp>
```

### Static Public Member Functions

- static bool **return\_path** ()

### 6.82.1 Detailed Description

```
template<int UPPER>
struct util::path_type::RANDOM_PATH< UPPER >
```

Finds a random path by returning true only when a random number in the range (0, UPPER) is equal to 0.

(UPPER is specialized as a template parameter.)

Definition at line 169 of file PathfindingAlgorithms.hpp.

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

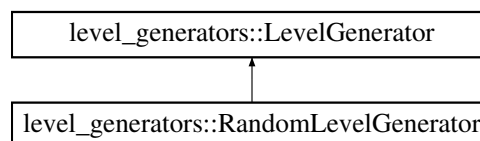
- tools/PathfindingAlgorithms.hpp

## 6.83 level\_generators::RandomLevelGenerator Class Reference

Level generator that uses simple RNG approach (counts the number of gold neighbours and increases the chance to spawn a gold deposit if needed).

```
#include <LevelGenerators.hpp>
```

Inheritance diagram for level\_generators::RandomLevelGenerator:



### Public Member Functions

- [RandomLevelGenerator](#) ([EntitySystem](#) &, tdt::uint)  
*Constructor.*
- void [generate](#) (tdt::uint, tdt::uint, [WaveSystem](#) &) override  
*Generates a level with the given dimensions using an RNG approach.*

### Additional Inherited Members

#### 6.83.1 Detailed Description

Level generator that uses simple RNG approach (counts the number of gold neighbours and increases the chance to spawn a gold deposit if needed).

Definition at line 60 of file LevelGenerators.hpp.

#### 6.83.2 Constructor & Destructor Documentation

6.83.2.1 level\_generators::RandomLevelGenerator::RandomLevelGenerator ( [EntitySystem](#) & *ents*, tdt::uint *c* )

Constructor.

## Parameters

<i>Entity</i>	system that contains the level's entities.
<i>Number</i>	of iterations done while generating.

Definition at line 14 of file LevelGenerators.cpp.

### 6.83.3 Member Function Documentation

6.83.3.1 `void level_generators::RandomLevelGenerator::generate ( tdt::uint width, tdt::uint height, WaveSystem & wsystem ) [override],[virtual]`

Generates a level with the given dimensions using an RNG approach.

## Parameters

<i>Width</i>	of the level.
<i>Height</i>	of the level.
<i>Wave</i>	system that will have it's spawn nodes set.

0 == free space 1 == wall 2 == gold deposit 3 == border 4 == walkway 5 == light source 6 == throne 7 == vault 8 == mine

Implements [level\\_generators::LevelGenerator](#).

Definition at line 18 of file LevelGenerators.cpp.

The documentation for this class was generated from the following files:

- tools/LevelGenerators.hpp
- tools/LevelGenerators.cpp

## 6.84 RayCaster Class Reference

Manages polygon precise raycasting used with half walls that have empty spaces in their bounding boxes.

```
#include <RayCaster.hpp>
```

### Public Member Functions

- [RayCaster](#) (Ogre::SceneManager &)  
*Constructor.*
- `std::pair< bool, tdt::real > cast (const Ogre::Vector3 &, const Ogre::Vector3 &, const std::string &= "") const`  
*Casts a ray that checks for polygon level collisions on it's way.*

## Private Member Functions

- void [get\\_info](#) (const Ogre::Entity &, tdt::uint &, tdt::uint &, std::vector< Ogre::Vector3 > &, std::vector< tdt::uint > &, const Ogre::Vector3 &, const Ogre::Quaternion &, const Ogre::Vector3 &) const

*Returns information about vertices of an entity by modifying it's size\_t and vector parameters.*

## Private Attributes

- Ogre::RaySceneQuery \* [query\\_](#)

*Query used for the collision ray cast.*

### 6.84.1 Detailed Description

Manages polygon precise raycasting used with half walls that have empty spaces in their bounding boxes.

#### Note

Strongly inspired by <http://www.ogre3d.org/tikiwiki/Raycasting+to+the+polygon+level> from the official Ogre3D wiki, big thanks to all contributors.

Definition at line 14 of file RayCaster.hpp.

### 6.84.2 Constructor & Destructor Documentation

#### 6.84.2.1 RayCaster::RayCaster ( Ogre::SceneManager & mgr )

Constructor.

##### Parameters

<i>Scene</i>	manager that is used to create the ray query.
--------------	---

Definition at line 6 of file RayCaster.cpp.

### 6.84.3 Member Function Documentation

#### 6.84.3.1 std::pair< bool, tdt::real > RayCaster::cast ( const Ogre::Vector3 & start, const Ogre::Vector3 & dir, const std::string & target = " " ) const

Casts a ray that checks for polygon level collisions on it's way.

Returns a pair of a bool, signaling whether a hit was made and a distance to the place of the collision, if any occurred.

##### Parameters

<i>Starting</i>	position of the ray.
<i>Direction</i>	of the ray.
<i>Name</i>	of the entity to be ignored (the target) if it's a wall.

Definition at line 13 of file RayCaster.cpp.

```
6.84.3.2 void RayCaster::get_info ( const Ogre::Entity & ent, tdt::uint & v_count, tdt::uint & i_count, std::vector<
      Ogre::Vector3 > & verts, std::vector< tdt::uint > & inds, const Ogre::Vector3 & position, const Ogre::Quaternion &
      orientation, const Ogre::Vector3 & scale ) const [private]
```

Returns information about vertices of an entity by modifying it's size\_t and vector parameters.

#### Parameters

<i>Entity</i>	to be checked.
<i>Number</i>	of vertices, will be set inside the function.
<i>Number</i>	of indices, will be set inside the function.
<i>Vector</i>	of vertex point positions, will be filled inside the function.
<i>Vector</i>	of indices of the vertex points inside the vertex vector above.
<i>Position</i>	of the entity.
<i>Orientation</i>	of the entity.
<i>Scale</i>	of the entity.

Definition at line 73 of file RayCaster.cpp.

## 6.84.4 Member Data Documentation

```
6.84.4.1 Ogre::RaySceneQuery* RayCaster::query_ [private]
```

Query used for the collision ray cast.

Definition at line 38 of file RayCaster.hpp.

The documentation for this class was generated from the following files:

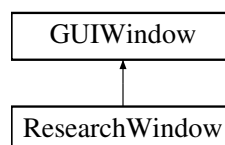
- tools/RayCaster.hpp
- tools/RayCaster.cpp

## 6.85 ResearchWindow Class Reference

Class that represents the research window in the game, which allows the player to unlock new buildings and spells.

```
#include <ResearchWindow.hpp>
```

Inheritance diagram for ResearchWindow:



## Public Member Functions

- [ResearchWindow](#) ()  
*Constructor.*
- [~ResearchWindow](#) ()=default  
*Destructor.*
- void [unlock](#) (tdt::uint, tdt::uint)  
*Unlocks a single research point at a given position in the research table.*
- void [dummy\\_unlock](#) (tdt::uint, tdt::uint)  
*Unlocks a single research point without activating it.*
- const std::array< bool, 42 > & [get\\_unlocked](#) () const  
*Returns a reference to the unlock table, used for serialization.*
- void [show](#) (tdt::uint, tdt::uint, bool=true)  
*Shows a single research point at a given position in the research table.*
- void [free\\_research](#) ()  
*Cheat that changes the price of any research to 0.*
- void [research\\_all](#) ()  
*Cheat that unlocks all research points.*
- void [reset\\_research](#) ()  
*Resets the research state so that all items can be unlocked again.*

## Protected Member Functions

- void [init\\_](#) () override  
*Initializes this window.*

## Private Member Functions

- tdt::uint [get\\_price\\_](#) (tdt::uint, tdt::uint)  
*Returns the price in gold of a research point at the given position in the research table.*
- bool [is\\_unlocked\\_](#) (tdt::uint, tdt::uint)  
*Returns true if the research point at the given position is unlocked, false otherwise.*

## Private Attributes

- [lpp::Script](#) \* [script\\_](#)  
*Pointer to the Lua Script used for easier access.*
- const tdt::uint [rows\\_](#) {6}  
*Number of rows that the research table has.*
- const tdt::uint [cols\\_](#) {7}  
*Number of columns that the research table has.*
- std::array< tdt::uint, 42 > [prices\\_](#)  
*Contains prices of the individual research points.*
- std::array< bool, 42 > [unlocked\\_](#)  
*Contains information about the lock status of the individual research points.*

## Friends

- class [GameSerializer](#)

## Additional Inherited Members

### 6.85.1 Detailed Description

Class that represents the research window in the game, which allows the player to unlock new buildings and spells.

#### Note

Because this class is so tightly bound to Lua, the indices have to be adjusted when accessing the `prices_` and `unlocked_` arrays, since Lua uses indices starting at one when handling arrays.

Definition at line 18 of file `ResearchWindow.hpp`.

### 6.85.2 Constructor & Destructor Documentation

#### 6.85.2.1 `ResearchWindow::ResearchWindow ( )`

Constructor.

Definition at line 6 of file `ResearchWindow.cpp`.

#### 6.85.2.2 `ResearchWindow::~~ResearchWindow ( )` `[default]`

Destructor.

### 6.85.3 Member Function Documentation

#### 6.85.3.1 `void ResearchWindow::dummy_unlock ( tdt::uint i, tdt::uint j )`

Unlocks a single research point without activating it.

Used for serialization.

#### Parameters

<i>Row</i>	number.
<i>Column</i>	number.

Definition at line 28 of file `ResearchWindow.cpp`.

#### 6.85.3.2 `void ResearchWindow::free_research ( )`

Cheat that changes the price of any research to 0.

Definition at line 51 of file `ResearchWindow.cpp`.



**6.85.3.3** `tdt::uint ResearchWindow::get_price_( tdt::uint i, tdt::uint j ) [private]`

Returns the price in gold of a research point at the given position in the research table.

Parameters

<i>Row</i>	number.
<i>Column</i>	number.

Definition at line 154 of file ResearchWindow.cpp.

**6.85.3.4** `const std::array< bool, 42 > & ResearchWindow::get_unlocked ( ) const`

Returns a reference to the unlock table, used for serialization.

Definition at line 41 of file ResearchWindow.cpp.

**6.85.3.5** `void ResearchWindow::init_ ( ) [override],[protected],[virtual]`

Initializes this window.

Makes sure all buttons can be used in the beggining.

Research button initialization.

Note

AlfiskoSkin does not have tooltip support, maybe create own skin? `butt->setTooltipText( script_->call<std::string, tdt::uint, tdt::uint>( "game.gui.research.get_tooltip", i, j ) );`

Implements [GUIWindow](#).

Definition at line 95 of file ResearchWindow.cpp.

**6.85.3.6** `bool ResearchWindow::is_unlocked_ ( tdt::uint i, tdt::uint j ) [private]`

Returns true if the research point at the given position is unlocked, false otherwise.

Parameters

<i>Row</i>	number.
<i>Column</i>	number.

Definition at line 163 of file ResearchWindow.cpp.

**6.85.3.7** `void ResearchWindow::research_all ( )`

Cheat that unlocks all research points.

Definition at line 57 of file ResearchWindow.cpp.

#### 6.85.3.8 void ResearchWindow::reset\_research ( )

Resets the research state so that all items can be unlocked again.

Definition at line 68 of file ResearchWindow.cpp.

#### 6.85.3.9 void ResearchWindow::show ( tdt::uint *i*, tdt::uint *j*, bool *val* = true )

Shows a single research point at a given position in the research table.

##### Parameters

<i>Row</i>	number.
<i>Column</i>	number.
<i>If</i>	true, shows the button, otherwise it hides it.

Definition at line 46 of file ResearchWindow.cpp.

#### 6.85.3.10 void ResearchWindow::unlock ( tdt::uint *i*, tdt::uint *j* )

Unlocks a single research point at a given position in the research table.

##### Parameters

<i>Row</i>	number.
<i>Column</i>	number.

Definition at line 10 of file ResearchWindow.cpp.

### 6.85.4 Member Data Documentation

#### 6.85.4.1 const tdt::uint ResearchWindow::cols\_{7} [private]

Number of columns that the research table has.

Definition at line 115 of file ResearchWindow.hpp.

#### 6.85.4.2 std::array<tdt::uint, 42> ResearchWindow::prices\_ [private]

Contains prices of the individual research points.

Used to avoid unnecessary Lua lookups.

Definition at line 121 of file ResearchWindow.hpp.

#### 6.85.4.3 `const tdt::uint ResearchWindow::rows_ {6} [private]`

Number of rows that the research table has.

Definition at line 110 of file ResearchWindow.hpp.

#### 6.85.4.4 `lpp::Script* ResearchWindow::script_ [private]`

Pointer to the Lua Script used for easier access.

Definition at line 105 of file ResearchWindow.hpp.

#### 6.85.4.5 `std::array<bool, 42> ResearchWindow::unlocked_ [private]`

Contains information about the lock status of the individual research points.

Used to avoid unnecessary Lua lookups.

Definition at line 128 of file ResearchWindow.hpp.

The documentation for this class was generated from the following files:

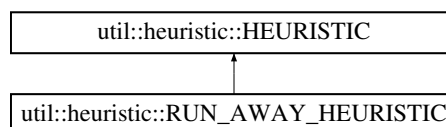
- gui/ResearchWindow.hpp
- gui/ResearchWindow.cpp

## 6.86 util::heuristic::RUN\_AWAY\_HEURISTIC Struct Reference

Used by entities that want to run away from an enemy.

```
#include <PathfindingAlgorithms.hpp>
```

Inheritance diagram for util::heuristic::RUN\_AWAY\_HEURISTIC:



### Public Member Functions

- **RUN\_AWAY\_HEURISTIC** ([EntitySystem](#) &ents, tdt::uint from)
- tdt::real **get\_cost** (tdt::uint id1, tdt::uint id2) override

### Private Attributes

- tdt::uint **from\_**

## Additional Inherited Members

### 6.86.1 Detailed Description

Used by entities that want to run away from an enemy.

Definition at line 240 of file PathfindingAlgorithms.hpp.

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

- tools/PathfindingAlgorithms.hpp

## 6.87 Ipp::Script Class Reference

Class representing a Lua script, allows to register C++ functions, load variables, call functions, execute strings containing Lua code and other functionalities.

```
#include <IppScript.hpp>
```

### Public Types

- using **state** = lua\_State \*
- using **regs** = luaL\_Reg

### Public Member Functions

- [Script](#) (const [Script](#) &)=delete  
*Copying this script might cause the Lua state get closed when one of the copies gets destroyed and would cause the game to be unable to use it's scripting engine (and thus crashing probably).*
- [~Script](#) ()  
*Destructor, closes the Lua virtual machine.*
- state [get\\_state](#) ()  
*Returns the lua state representing the Lua virtual machine.*
- void [execute](#) (const std::string &)  
*Executes a given string from within Lua.*
- void [register\\_function](#) (const std::string &, lua\_CFunction)  
*Registers a C++ function which can then be used from within Lua.*
- void [load](#) (const std::string &)  
*Loads, compiles and executes a Lua script.*
- bool [is\\_nil](#) (const std::string &)  
*Returns true if a given value is nil, false otherwise.*
- template<typename T >  
T [get](#) (const std::string &name)  
*Retrieves and returns a value from Lua.*
- template<typename Result , typename... Args>  
Result [call](#) (const std::string &fname, Args...as)  
*Calls a given Lua function.*

- `template<typename Result >`  
`Result call (const std::string &fname)`  
*Calls a given Lua function.*
- `template<typename T >`  
`void set (const std::string &name, T val)`  
*Sets a given variable to a given value.*
- `template<typename T >`  
`std::vector< T > get_vector (const std::string &name)`  
*Retrieves a Lua array table (integer indexing) in the form of a C++ vector.*
- `std::string get_stack_contents ()`  
*Returns string representation of the Lua stack.*
- `void reload_all_scripts ()`  
*Reloads all script files that have been previously loaded.*
- `template<>`  
`void set (const std::string &name, bool val)`

### Static Public Member Functions

- `static Script & instance ()`  
*Returns a reference to the `lpp::Script` singleton.*

### Private Member Functions

- `Script ()`  
*Constructor, kept private because of the use of the singleton pattern.*
- `std::string get_field_to_stack (const std::string &)`  
*Gets a nested value (inside a table hierarchy) on top of the stack and returns the name of the final variable (without table prefixes).*
- `void clear_stack ()`  
*Pops everything off the stack.*
- `template<typename T >`  
`T get_ (const std::string &name="unknown")`  
*Returns the value stored on top of the stack.*
- `template<typename Arg , typename... Args>`  
`int push_args (Arg a, Args...as)`  
*Pushed a variadic list of arguments onto the stack to be passed as arguments to a Lua function call, returns the amount of arguments pushed onto the stack.*
- `template<typename Arg >`  
`int push_args (Arg a)`  
*Bottom case of the push\_args recursive call.*
- `template<typename Arg >`  
`void push_arg (Arg a)`  
*Pushes a single value onto the Lua stack.*
- `template<>`  
`int get_ (const std::string &name)`
- `template<>`  
`bool get_ (const std::string &name)`
- `template<>`  
`void get_ (const std::string &name)`
- `template<>`  
`void push_arg (int arg)`  
*Specializations of the method `lpp::Script::push_arg`, which pushes a single function argument onto the Lua stack.*
- `template<>`  
`void push_arg (float arg)`
- `template<>`  
`void push_arg (bool arg)`

## Private Attributes

- state [L](#)  
*Lua state representing the Lua virtual machine.*
- `std::set< std::string >` [loaded\\_scripts\\_](#)  
*Contains the names of all scripts loaded during the current runtime.*

### 6.87.1 Detailed Description

Class representing a Lua script, allows to register C++ functions, load variables, call functions, execute strings containing Lua code and other functionalities.

Definition at line 18 of file `LppScript.hpp`.

### 6.87.2 Constructor & Destructor Documentation

#### 6.87.2.1 `lpp::Script::Script ( const Script & )` `[delete]`

Copying this script might cause the Lua state get closed when one of the copies gets destroyed and would cause the game to be unable to use it's scripting engine (and thus crashing probably).

#### 6.87.2.2 `lpp::Script::~Script ( )` `[inline]`

Destructor, closes the Lua virtual machine.

Definition at line 34 of file `LppScript.hpp`.

#### 6.87.2.3 `lpp::Script::Script ( )` `[private]`

Constructor, kept private because of the use of the singleton pattern.

[lpp::Script](#) definitions:

Definition at line 7 of file `LppScript.cpp`.

### 6.87.3 Member Function Documentation

#### 6.87.3.1 `template<typename Result , typename... Args> Result lpp::Script::call ( const std::string & fname, Args... as )` `[inline]`

Calls a given Lua function.

#### Parameters

<i>Name</i>	of the function.
<i>Variadic</i>	list of arguments that are passed to the function.

Definition at line 99 of file LppScript.hpp.

**6.87.3.2** `template<typename Result > Result lpp::Script::call ( const std::string & fname ) [inline]`

Calls a given Lua function.

#### Parameters

<i>Name</i>	of the function.
-------------	------------------

Definition at line 121 of file LppScript.hpp.

**6.87.3.3** `void lpp::Script::clear_stack ( ) [private]`

Pops everything off the stack.

Definition at line 66 of file LppScript.cpp.

**6.87.3.4** `void lpp::Script::execute ( const std::string & command )`

Executes a given string from within Lua.

#### Parameters

<i>String</i>	containing commands to be executed.
---------------	-------------------------------------

Definition at line 14 of file LppScript.cpp.

**6.87.3.5** `template<typename T > T lpp::Script::get ( const std::string & name ) [inline]`

Retrieves and returns a value from Lua.

#### Parameters

<i>Name</i>	of the variable containing the desired value.
-------------	---

Definition at line 74 of file LppScript.hpp.

**6.87.3.6** `template<typename T > T lpp::Script::get_ ( const std::string & name = "unknown" ) [private]`

Returns the value stored on top of the stack.

**6.87.3.7** `std::string lpp::Script::get_field_to_stack ( const std::string & name )` `[private]`

Gets a nested value (inside a table hierarchy) on top of the stack and returns the name of the final variable (without table prefixes).



## Parameters

<i>Full</i>	name of the variable.
-------------	-----------------------

Definition at line 47 of file LppScript.cpp.

#### 6.87.3.8 std::string lpp::Script::get\_stack\_contents ( )

Returns string representation of the Lua stack.

Definition at line 72 of file LppScript.cpp.

#### 6.87.3.9 state lpp::Script::get\_state ( ) [inline]

Returns the lua state representing the Lua virtual machine.

Definition at line 39 of file LppScript.hpp.

#### 6.87.3.10 template<typename T> std::vector<T> lpp::Script::get\_vector ( const std::string & name ) [inline]

Retrieves a Lua array table (integer indexing) in the form of a C++ vector.

## Parameters

<i>Name</i>	of the array.
-------------	---------------

Definition at line 153 of file LppScript.hpp.

#### 6.87.3.11 lpp::Script & lpp::Script::instance ( ) [static]

Returns a reference to the [lpp::Script](#) singleton.

Definition at line 105 of file LppScript.cpp.

#### 6.87.3.12 bool lpp::Script::is\_nil ( const std::string & name )

Returns true if a given value is nil, false otherwise.

## Parameters

<i>Name</i>	of the variable containing the desired value.
-------------	---

Definition at line 34 of file LppScript.cpp.

#### 6.87.3.13 void lpp::Script::load ( const std::string & fname )

Loads, compiles and executes a Lua script.

##### Parameters

<i>Name</i>	of the script file.
-------------	---------------------

Definition at line 25 of file LppScript.cpp.

#### 6.87.3.14 template<typename Arg > void lpp::Script::push\_arg ( Arg a ) [inline],[private]

Pushes a single value onto the Lua stack.

##### Parameters

<i>Value</i>	to be pushed.
--------------	---------------

##### Note

By default does nothing, only specialised versions push anything.

Definition at line 250 of file LppScript.hpp.

#### 6.87.3.15 template<> void lpp::Script::push\_arg ( int arg ) [inline],[private]

Specializations of the method [lpp::Script::push\\_arg](#), which pushes a single function argument onto the Lua stack.

Definition at line 370 of file LppScript.hpp.

#### 6.87.3.16 template<typename Arg , typename... Args> int lpp::Script::push\_args ( Arg a, Args... as ) [inline],[private]

Pushed a variadic list of arguments onto the stack to be passed as arguments to a Lua function call, returns the amount of arguments pushed onto the stack.

##### Parameters

<i>First</i>	argument in the list.
<i>Tail</i>	argument list used in recursive call.

Definition at line 226 of file LppScript.hpp.

#### 6.87.3.17 template<typename Arg > int lpp::Script::push\_args ( Arg a ) [inline],[private]

Bottom case of the push\_args recursive call.

## Parameters

<i>Argument</i>	to be pushed onto the stack.
-----------------	------------------------------

Definition at line 237 of file LppScript.hpp.

**6.87.3.18** `void lpp::Script::register_function ( const std::string & name, lua_CFunction fn )`

Registers a C++ function which can then be used from within Lua.

## Parameters

<i>Name</i>	of the function.
<i>Function</i>	to be registered, it has to have the signature <code>int fname(lua_State*)</code> and return the number of results pushed onto the stack, arguments are on the stack.

Definition at line 20 of file LppScript.cpp.

**6.87.3.19** `void lpp::Script::reload_all_scripts ( )`

Reloads all script files that have been previously loaded.

Definition at line 99 of file LppScript.cpp.

**6.87.3.20** `template<typename T> void lpp::Script::set ( const std::string & name, T val )` `[inline]`

Sets a given variable to a given value.

## Parameters

<i>Variable</i>	to be changed.
<i>Value</i>	that the variable should be changed to.

Definition at line 142 of file LppScript.hpp.

## 6.87.4 Member Data Documentation

**6.87.4.1** `state lpp::Script::L` `[private]`

Lua state representing the Lua virtual machine.

Definition at line 250 of file LppScript.hpp.

#### 6.87.4.2 `std::set<std::string> lpp::Script::loaded_scripts_` [private]

Contains the names of all scripts loaded during the current runtime.

Definition at line 260 of file `LppScript.hpp`.

The documentation for this class was generated from the following files:

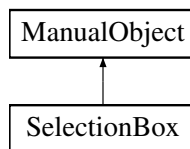
- `lppscript/LppScript.hpp`
- `lppscript/LppScript.cpp`

## 6.88 SelectionBox Class Reference

Class representing the ingame selection box (created by moving the left mouse while pressing the left mouse button) to select multiple entities on screen or a single entity (by simply clicking).

```
#include <SelectionBox.hpp>
```

Inheritance diagram for SelectionBox:



### Public Member Functions

- [SelectionBox](#) (const Ogre::String &, [EntitySystem](#) &, Ogre::PlaneBoundedVolumeListSceneQuery &, Ogre::RaySceneQuery &, Ogre::SceneManager &)  
*Constructor.*
- [~SelectionBox](#) ()  
*Destructor.*
- void [set\\_corners](#) (tdt::real, tdt::real, tdt::real, tdt::real)  
*Recreates the selection box with the given coordinates.*
- void [set\\_corners](#) (const Ogre::Vector2 &, const Ogre::Vector2 &)  
*Overload of the SelectionBox::set\_corners(float, float, float, float) method used as a simple interface when using vectors.*
- std::vector< tdt::uint > & [get\\_selected\\_entities](#) ()  
*Returns (by reference) a vector containing ID's of all currently selected entities.*
- void [select\\_object](#) (Ogre::MovableObject &)  
*Adds a given object to the vector of selected entities.*
- void [clear\\_selected\\_entities](#) ()  
*Deselects all currently selected entities.*
- void [execute\\_selection](#) (const Ogre::Vector2 &, Ogre::Camera &, bool=false)  
*Performs the selection, to be called when the mouse is released.*
- void [set\\_starting\\_point](#) (const Ogre::Vector2 &)  
*Sets the starting point of the current selection, should be called upon the initial mouse button click.*
- void [set\\_selecting](#) (bool)  
*Sets the selecting mode (and the visibility flag of the selection box).*
- bool [is\\_selecting](#) () const  
*Returns true if the selection box is currently selecting.*
- void [extend\\_to](#) (const Ogre::Vector2 &)  
*Extends the selection box to a given coordinate, using the start\_ coordinate provided by SelectionBox::set\_starting\_point as the initial point.*

## Private Member Functions

- void [execute\\_single\\_selection](#) (Ogre::Camera &)  
*Used within the [SelectionBox::execute\\_selection](#) method when the selection box is too small (i.e.*

## Private Attributes

- std::vector< tdt::uint > [selected\\_entities\\_](#)  
*Currently selected entities.*
- EntitySystem & [entities\\_](#)  
*Reference to the game's entity system, used to identify an entity ID from an entity SceneNode.*
- Ogre::Vector2 [start\\_](#)  
*Vector containing the coordinates of the starting point of the selection.*
- Ogre::PlaneBoundedVolumeListSceneQuery & [volume\\_query\\_](#)  
*Scene query used to find all the entities that are within the selection box.*
- Ogre::RaySceneQuery & [ray\\_query\\_](#)  
*Scene query used to find an entity when the selection box is too small.*
- bool [selection\\_in\\_progress\\_](#)  
*Determines if the player is currently selecting entities.*
- Ogre::SceneManager & [scene\\_mgr\\_](#)  
*Reference to the game's main scene manager.*

### 6.88.1 Detailed Description

Class representing the ingame selection box (created by moving the left mouse while pressing the left mouse button) to select multiple entities on screen or a single entity (by simply clicking).

Definition at line 13 of file SelectionBox.hpp.

### 6.88.2 Constructor & Destructor Documentation

**6.88.2.1** SelectionBox::SelectionBox ( const Ogre::String & *name*, EntitySystem & *ents*, Ogre::PlaneBoundedVolumeListSceneQuery & *vol\_query*, Ogre::RaySceneQuery & *ray\_query*, Ogre::SceneManager & *mgr* )

Constructor.

Parameters

<i>Name</i>	of this object (passed to ManualObject constructor - hence the type).
<i>Reference</i>	to a volume query used for multi selection.
<i>Reference</i>	to a ray query used for single selection.
<i>Reference</i>	to the game's main scene manager.

Definition at line 5 of file SelectionBox.cpp.

### 6.88.2.2 `SelectionBox::~~SelectionBox ( )`

Destructor.

#### Note

Even thous the queries are captured by reference, they need to be destroyed manually using the scene manager.

Definition at line 19 of file `SelectionBox.cpp`.

## 6.88.3 Member Function Documentation

### 6.88.3.1 `void SelectionBox::clear_selected_entities ( )`

Deselects all currently selected entities.

Definition at line 76 of file `SelectionBox.cpp`.

### 6.88.3.2 `void SelectionBox::execute_selection ( const Ogre::Vector2 & end, Ogre::Camera & cam, bool append = false )`

Performs the selection, to be called when the mouse is released.

#### Parameters

<i>Coordinate</i>	of the end point of the box (mouse position).
<i>Reference</i>	to the camera the query should be performed from.
<i>True</i>	if the the new selection should be appended to the current selection.

#### Note

The starting point has to be saved prior to this call (i.e. when the mouse button gets pressed) in order for the selection to work properly.

Definition at line 88 of file `SelectionBox.cpp`.

### 6.88.3.3 `void SelectionBox::execute_single_selection ( Ogre::Camera & cam ) [private]`

Used within the [SelectionBox::execute\\_selection](#) method when the selection box is too small (i.e. a single click) and performs a ray query selecting a single closest entity under the player's cursor.

#### Parameters

<i>Reference</i>	to the game's main camera.
------------------	----------------------------

Definition at line 144 of file SelectionBox.cpp.

#### 6.88.3.4 void SelectionBox::extend\_to ( const Ogre::Vector2 & end )

Extends the selection box to a given coordinate, using the start\_ coordinate provided by [SelectionBox::set\\_starting\\_point](#) as the initial point.

##### Parameters

<i>Coordinate</i>	of the ending point.
-------------------	----------------------

Definition at line 139 of file SelectionBox.cpp.

#### 6.88.3.5 std::vector< tdt::uint > & SelectionBox::get\_selected\_entities ( )

Returns (by reference) a vector containing ID's of all currently selected entities.

Definition at line 57 of file SelectionBox.cpp.

#### 6.88.3.6 bool SelectionBox::is\_selecting ( ) const

Returns true if the selection box is currently selecting.

Definition at line 134 of file SelectionBox.cpp.

#### 6.88.3.7 void SelectionBox::select\_object ( Ogre::MovableObject & obj )

Adds a given object to the vector of selected entities.

##### Parameters

<i>Reference</i>	to the object from which the entity ID will be deduced.
------------------	---

Definition at line 62 of file SelectionBox.cpp.

#### 6.88.3.8 void SelectionBox::set\_corners ( tdt::real left, tdt::real top, tdt::real right, tdt::real bott )

Recreates the selection box with the given coordinates.

##### Parameters

<i>Left</i>	side coordinate.
<i>Top</i>	side coordinate.
<i>Right</i>	side coordinate.
<i>Bottom</i>	axis coordinate.

Necessary translation, mouse positions are normalized to belong to the (0,1) range, but the `Ogre::ManualObject` creation requires them to belong to the (-1, 1).

Definition at line 25 of file `SelectionBox.cpp`.

#### 6.88.3.9 void SelectionBox::set\_corners ( const Ogre::Vector2 & t\_l, const Ogre::Vector2 & b\_r )

Overload of the `SelectionBox::set_corners(float, float, float, float)` method used as a simple interface when using vectors.

##### Parameters

<i>Coordinate</i>	of the top left corner.
<i>Coordinate</i>	of the bottom right corner.

Definition at line 50 of file `SelectionBox.cpp`.

#### 6.88.3.10 void SelectionBox::set\_selecting ( bool sel )

Sets the selecting mode (and the visibility flag of the selection box).

##### Parameters

<i>Bool</i>	value representing the new visibility mode.
-------------	---

Definition at line 128 of file `SelectionBox.cpp`.

#### 6.88.3.11 void SelectionBox::set\_starting\_point ( const Ogre::Vector2 & start )

Sets the starting point of the current selection, should be called upon the initial mouse button click.

##### Parameters

<i>Coordinate</i>	of the starting point.
-------------------	------------------------

Definition at line 123 of file `SelectionBox.cpp`.

## 6.88.4 Member Data Documentation

#### 6.88.4.1 EntitySystem& SelectionBox::entities\_ [private]

Reference to the game's entity system, used to identify an entity ID from an entity `SceneNode`.

Definition at line 120 of file `SelectionBox.hpp`.



#### 6.88.4.2 `Ogre::RaySceneQuery& SelectionBox::ray_query_` [private]

Scene query used to find an entity when the selection box is too small.

Definition at line 136 of file SelectionBox.hpp.

#### 6.88.4.3 `Ogre::SceneManager& SelectionBox::scene_mgr_` [private]

Reference to the game's main scene manager.

Definition at line 146 of file SelectionBox.hpp.

#### 6.88.4.4 `std::vector<tdt::uint> SelectionBox::selected_entities_` [private]

Currently selected entities.

Definition at line 114 of file SelectionBox.hpp.

#### 6.88.4.5 `bool SelectionBox::selection_in_progress_` [private]

Determines if the player is currently selecting entities.

Definition at line 141 of file SelectionBox.hpp.

#### 6.88.4.6 `Ogre::Vector2 SelectionBox::start_` [private]

Vector containing the coordinates of the starting point of the selection.

(Ending point will be specified in the [SelectionBox::execute\\_selection](#) method.)

Definition at line 126 of file SelectionBox.hpp.

#### 6.88.4.7 `Ogre::PlaneBoundedVolumeListSceneQuery& SelectionBox::volume_query_` [private]

Scene query used to find all the entities that are within the selection box.

Definition at line 131 of file SelectionBox.hpp.

The documentation for this class was generated from the following files:

- tools/SelectionBox.hpp
- tools/SelectionBox.cpp

## 6.89 Spellcaster::SPELL Struct Reference

A structure representing a spell by containing it's type and name.

## Public Attributes

- SPELL\_TYPE **type\_**
- std::string **spell\_**

### 6.89.1 Detailed Description

A structure representing a spell by containing it's type and name.

Definition at line 102 of file Spellcaster.hpp.

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

- tools/Spellcaster.hpp

## 6.90 Spellcaster Class Reference

A utility class that manages the player's spell casting and is usually called from input handlers and the spell casting window.

```
#include <Spellcaster.hpp>
```

## Classes

- struct [SPELL](#)  
*A structure representing a spell by containing it's type and name.*

## Public Member Functions

- [Spellcaster](#) ([EntityPlacer](#) &, [SelectionBox](#) &)  
*Constructor.*
- [~Spellcaster](#) ()=default  
*Destructor.*
- void [set\\_spell\\_type](#) (SPELL\_TYPE)  
*Sets the type of the currently casted spell.*
- SPELL\_TYPE [get\\_spell\\_type](#) () const  
*Returns the type of the currently casted spell.*
- void [set\\_spell](#) (const std::string &)  
*Sets the name of the currently casted spell.*
- const std::string & [get\\_spell](#) () const  
*Returns the name of the currently casted spell.*
- void [cast](#) (Ogre::Vector2=Ogre::Vector2{})  
*Applies the effect of the currently casted spell.*
- SPELL\_TYPE [get\\_last\\_spell\\_type](#) () const  
*Returns the type of the previously casted spell.*
- const std::string & [get\\_last\\_spell](#) () const  
*Returns the name of the previously casted spell.*
- void [set\\_last\\_spell\\_id](#) (tdt::uint)  
*Sets the ID of the entity created by the last spell.*
- tdt::uint [get\\_last\\_spell\\_id](#) () const  
*Returns the ID of the entity created by the last spell.*
- bool [is\\_casting](#) () const  
*Returns true if this spellcaster is currently casting, false otherwise.*
- void [stop\\_casting](#) ()  
*Immediately stops any cast being performed.*

## Private Attributes

- [EntityPlacer](#) & [placer\\_](#)  
*Used to place entities created by placing spell.*
- [SelectionBox](#) & [selector\\_](#)  
*Used to get selected targets for targeted spells.*
- [lpp::Script](#) & [script\\_](#)  
*Reference to the scripting engine used for easier calls to the spell functions.*
- [SPELL curr\\_spell\\_](#)  
*The spell that is currently being casted.*
- [SPELL last\\_spell\\_](#)  
*The spell that has been casted previously.*
- [tdt::uint](#) [last\\_spell\\_id\\_](#)  
*ID of the entity created by a placing spell (if any).*

### 6.90.1 Detailed Description

A utility class that manages the player's spell casting and is usually called from input handlers and the spell casting window.

Definition at line 18 of file Spellcaster.hpp.

### 6.90.2 Constructor & Destructor Documentation

#### 6.90.2.1 Spellcaster::Spellcaster ( [EntityPlacer](#) & *placer*, [SelectionBox](#) & *selector* )

Constructor.

Parameters

<i>The</i>	placer that is used for placing spells.
<i>Selector</i>	that is used for targeted spells.

Definition at line 8 of file Spellcaster.cpp.

#### 6.90.2.2 Spellcaster::~Spellcaster ( ) [default]

Destructor.

### 6.90.3 Member Function Documentation

#### 6.90.3.1 void Spellcaster::cast ( [Ogre::Vector2](#) *mouse\_position* = [Ogre::Vector2](#){ } )

Applies the effect of the currently casted spell.

## Parameters

<i>Optional</i>	mouse position parameter for positional spells.
-----------------	---

Definition at line 36 of file Spellcaster.cpp.

#### 6.90.3.2 `const std::string & Spellcaster::get_last_spell ( ) const`

Returns the name of the previously casted spell.

(Used if spells have sequential effect - like portals.)

Definition at line 83 of file Spellcaster.cpp.

#### 6.90.3.3 `tdt::uint Spellcaster::get_last_spell_id ( ) const`

Returns the ID of the entity created by the last spell.

(Used if spells have sequential effect - like portals.)

Definition at line 93 of file Spellcaster.cpp.

#### 6.90.3.4 `SPELL_TYPE Spellcaster::get_last_spell_type ( ) const`

Returns the type of the previously casted spell.

(Used if spells have sequential effect - like portals.)

Definition at line 78 of file Spellcaster.cpp.

#### 6.90.3.5 `const std::string & Spellcaster::get_spell ( ) const`

Returns the name of the currently casted spell.

Definition at line 31 of file Spellcaster.cpp.

#### 6.90.3.6 `SPELL_TYPE Spellcaster::get_spell_type ( ) const`

Returns the type of the currently casted spell.

Definition at line 20 of file Spellcaster.cpp.

#### 6.90.3.7 `bool Spellcaster::is_casting ( ) const`

Returns true if this spellcaster is currently casting, false otherwise.

Definition at line 98 of file Spellcaster.cpp.

#### 6.90.3.8 `void Spellcaster::set_last_spell_id ( tdt::uint val )`

Sets the ID of the entity created by the last spell.

(Used if spells have sequential effect - like portals.)

## Parameters

<i>The</i>	ID of the entity created.
------------	---------------------------

Definition at line 88 of file Spellcaster.cpp.

#### 6.90.3.9 void Spellcaster::set\_spell ( const std::string & val )

Sets the name of the currently casted spell.

## Parameters

<i>The</i>	new name.
------------	-----------

Definition at line 25 of file Spellcaster.cpp.

#### 6.90.3.10 void Spellcaster::set\_spell\_type ( SPELL\_TYPE val )

Sets the type of the currently casted spell.

## Parameters

<i>The</i>	new spell type.
------------	-----------------

Definition at line 14 of file Spellcaster.cpp.

#### 6.90.3.11 void Spellcaster::stop\_casting ( )

Immediately stops any cast being performed.

Definition at line 103 of file Spellcaster.cpp.

### 6.90.4 Member Data Documentation

#### 6.90.4.1 SPELL Spellcaster::curr\_spell\_ [private]

The spell that is currently being casted.

Definition at line 127 of file Spellcaster.hpp.

#### 6.90.4.2 SPELL Spellcaster::last\_spell\_ [private]

The spell that has been casted previously.

Definition at line 132 of file Spellcaster.hpp.

#### 6.90.4.3 `tdt::uint Spellcaster::last_spell_id_` [private]

ID of the entity created by a placing spell (if any).

Definition at line 137 of file Spellcaster.hpp.

#### 6.90.4.4 `EntityPlacer& Spellcaster::placer_` [private]

Used to place entities created by placing spell.

Definition at line 111 of file Spellcaster.hpp.

#### 6.90.4.5 `Ipp::Script& Spellcaster::script_` [private]

Reference to the scripting engine used for easier calls to the spell functions.

Definition at line 122 of file Spellcaster.hpp.

#### 6.90.4.6 `SelectionBox& Spellcaster::selector_` [private]

Used to get selected targets for targeted spells.

Definition at line 116 of file Spellcaster.hpp.

The documentation for this class was generated from the following files:

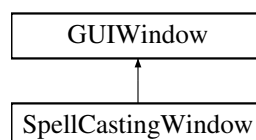
- tools/Spellcaster.hpp
- tools/Spellcaster.cpp

## 6.91 SpellCastingWindow Class Reference

Class representing the spell selection window, allows the player to cast registered (unlocked) spells.

```
#include <SpellCastingWindow.hpp>
```

Inheritance diagram for SpellCastingWindow:



## Public Member Functions

- [SpellCastingWindow](#) ()  
*Constructor.*
- [~SpellCastingWindow](#) ()=default  
*Destructor.*
- void [register\\_spell](#) (const std::string &)  
*Appends a table name to the vector of all spell tables.*
- void [set\\_caster](#) (Spellcaster \*)  
*Sets the caster instance used to cast the spells.*
- void [deactivate\\_current\\_spell](#) ()  
*Hides the "active" label.*
- const std::vector< std::string > & [get\\_spells](#) () const  
*Returns a vector containing the names of all unlocked spells.*
- void [clear\\_spells](#) ()  
*Removes all unlocked spell.*
- bool [dec\\_selection](#) ()  
*Decrements selection\_number\_ by one and updates the window.*
- bool [inc\\_selection](#) ()  
*Increments selection\_number\_ by one and updates the window.*
- void [set\\_spell\\_active](#) (int)  
*Marks a given spell as active.*
- void [cast](#) (int)  
*Casts a spell at a given position in the roster.*

## Protected Member Functions

- void [init\\_](#) () override  
*Initializes the window and subscribes events.*

## Private Member Functions

- const std::string & [get\\_spell\\_](#) (std::size\_t)  
*Range checked buildings\_ index access, returns the name of the building at a given index or "UNKNOWN" if the index is out of bounds.*
- void [update\\_selection\\_](#) ()  
*Updates building names on the buttons.*

## Private Attributes

- std::vector< std::string > [spells\\_](#)  
*Names of all registered buildings.*
- std::size\_t [selection\\_number\\_](#)  
*Number of the current rightmost selection.*
- lpp::Script \* [script\\_](#)  
*Pointer to the scripting engine used for easier access.*
- Spellcaster \* [caster\\_](#)  
*Does the actual spell casting once a spell is selected in this window.*
- int [curr\\_active\\_spell\\_](#)  
*Keeps track of the spell that is being currently casted.*

## Additional Inherited Members

### 6.91.1 Detailed Description

Class representing the spell selection window, allows the player to cast registered (unlocked) spells.

Definition at line 16 of file SpellCastingWindow.hpp.

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 SpellCastingWindow::SpellCastingWindow ( )

Constructor.

Definition at line 7 of file SpellCastingWindow.cpp.

#### 6.91.2.2 SpellCastingWindow::~SpellCastingWindow ( ) [default]

Destructor.

### 6.91.3 Member Function Documentation

#### 6.91.3.1 void SpellCastingWindow::cast ( int *spell\_num* )

Casts a spell at a given position in the roster.

##### Parameters

<i>The</i>	position of the spell (1-4).
------------	------------------------------

Definition at line 49 of file SpellCastingWindow.cpp.

#### 6.91.3.2 void SpellCastingWindow::clear\_spells ( )

Removes all unlocked spell.

Definition at line 38 of file SpellCastingWindow.cpp.

#### 6.91.3.3 void SpellCastingWindow::deactivate\_current\_spell ( )

Hides the "active" label.

Definition at line 27 of file SpellCastingWindow.cpp.



**6.91.3.4** `bool SpellCastingWindow::dec_selection ( )`

Decrements `selection_number_` by one and updates the window.

Definition at line 115 of file `SpellCastingWindow.cpp`.

**6.91.3.5** `const std::string & SpellCastingWindow::get_spell_ ( std::size_t index )` `[private]`

Range checked `buildings_` index access, returns the name of the building at a given index or "UNKNOWN" if the index is out of bounds.

**Parameters**

<i>Index</i>	of the building in the <code>buildings_</code> vector.
--------------	--

Definition at line 142 of file `SpellCastingWindow.cpp`.

**6.91.3.6** `const std::vector< std::string > & SpellCastingWindow::get_spells ( ) const`

Returns a vector containing the names of all unlocked spells.

(Used for serialization.)

Definition at line 33 of file `SpellCastingWindow.cpp`.

**6.91.3.7** `bool SpellCastingWindow::inc_selection ( )`

Increments `selection_number_` by one and updates the window.

Definition at line 129 of file `SpellCastingWindow.cpp`.

**6.91.3.8** `void SpellCastingWindow::init_ ( )` `[override]`, `[protected]`, `[virtual]`

Initializes the window and subscribes events.

Implements [GUIWindow](#).

Definition at line 61 of file `SpellCastingWindow.cpp`.

**6.91.3.9** `void SpellCastingWindow::register_spell ( const std::string & name )`

Appends a table name to the vector of all spell tables.

**Parameters**

<i>Name</i>	of the table to register.
-------------	---------------------------

Definition at line 11 of file SpellCastingWindow.cpp.

#### 6.91.3.10 void SpellCastingWindow::set\_caster ( Spellcaster \* caster )

Sets the caster instance used to cast the spells.

##### Parameters

<i>The</i>	new spell caster.
------------	-------------------

Definition at line 22 of file SpellCastingWindow.cpp.

#### 6.91.3.11 void SpellCastingWindow::set\_spell\_active ( int val )

Marks a given spell as active.

##### Parameters

<i>Position</i>	of the spell in the current view (0-3).
-----------------	---

Definition at line 182 of file SpellCastingWindow.cpp.

#### 6.91.3.12 void SpellCastingWindow::update\_selection\_ ( ) [private]

Updates building names on the buttons.

Definition at line 152 of file SpellCastingWindow.cpp.

### 6.91.4 Member Data Documentation

#### 6.91.4.1 Spellcaster\* SpellCastingWindow::caster\_ [private]

Does the actual spell casting once a spell is selected in this window.

Definition at line 124 of file SpellCastingWindow.hpp.

#### 6.91.4.2 int SpellCastingWindow::curr\_active\_spell\_ [private]

Keeps track of the spell that is being currently casted.

Definition at line 129 of file SpellCastingWindow.hpp.

**6.91.4.3** `lpp::Script* SpellCastingWindow::script_` [private]

Pointer to the scripting engine used for easier access.

Definition at line 118 of file `SpellCastingWindow.hpp`.

**6.91.4.4** `std::size_t SpellCastingWindow::selection_number_` [private]

Number of the current rightmost selection.

The window shows buildings with indices `<selection_number_ - 3, selection_number_>`.

Definition at line 112 of file `SpellCastingWindow.hpp`.

**6.91.4.5** `std::vector<std::string> SpellCastingWindow::spells_` [private]

Names of all registered buildings.

Definition at line 106 of file `SpellCastingWindow.hpp`.

The documentation for this class was generated from the following files:

- `gui/SpellCastingWindow.hpp`
- `gui/SpellCastingWindow.cpp`

## 6.92 SpellComponent Struct Reference

Allows an entity to periodically cast a spell.

```
#include <Components.hpp>
```

### Public Member Functions

- **SpellComponent** (`std::string &&b="ERROR", tdt::real cd=0.f`)
- **SpellComponent** (`const SpellComponent &)=default`
- **SpellComponent** (`SpellComponent &&)=default`
- `SpellComponent & operator=` (`const SpellComponent &)=default`
- `SpellComponent & operator=` (`SpellComponent &&)=default`

### Public Attributes

- `std::string blueprint`
- `tdt::real cd_time`
- `tdt::real cooldown`

## Static Public Attributes

- static constexpr int **type** = 10

### 6.92.1 Detailed Description

Allows an entity to periodically cast a spell.

Definition at line 272 of file Components.hpp.

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

- Components.hpp

## 6.93 StructureComponent Struct Reference

Defines a building (or a wall), by holding it's radius (of the area it takes in the grid) and vector of nodes that it sits on.

```
#include <Components.hpp>
```

## Public Member Functions

- **StructureComponent** (tdt::uint r=1, bool wt=false)
- **StructureComponent** (const [StructureComponent](#) &)=default
- **StructureComponent** ([StructureComponent](#) &&)=default
- [StructureComponent](#) & **operator=** (const [StructureComponent](#) &)=default
- [StructureComponent](#) & **operator=** ([StructureComponent](#) &&)=default

## Public Attributes

- tdt::uint **radius**
- bool **walk\_through**
- std::vector< tdt::uint > **residences**

## Static Public Attributes

- static constexpr int **type** = 17

### 6.93.1 Detailed Description

Defines a building (or a wall), by holding it's radius (of the area it takes in the grid) and vector of nodes that it sits on.

Definition at line 440 of file Components.hpp.

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

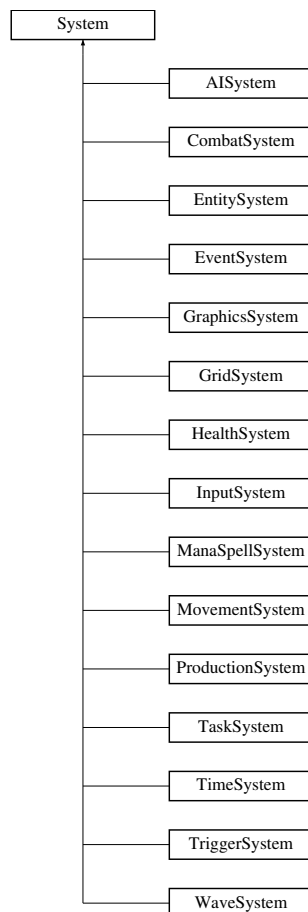
- Components.hpp

## 6.94 System Class Reference

Parent class of all systems.

```
#include <System.hpp>
```

Inheritance diagram for System:



### Public Member Functions

- virtual void [update](#) (tdt::real)=0  
*Updates the system.*
- virtual [~System](#) ()=default  
*Destructor.*

#### 6.94.1 Detailed Description

Parent class of all systems.

Definition at line 8 of file System.hpp.

## 6.94.2 Constructor & Destructor Documentation

### 6.94.2.1 virtual System::~System ( ) [virtual],[default]

Destructor.

## 6.94.3 Member Function Documentation

### 6.94.3.1 virtual void System::update ( tdt::real ) [pure virtual]

Updates the system.

Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implemented in [CombatSystem](#), [EntitySystem](#), [WaveSystem](#), [InputSystem](#), [TaskSystem](#), [ProductionSystem](#), [GridSystem](#), [TriggerSystem](#), [HealthSystem](#), [ManaSpellSystem](#), [AISystem](#), [GraphicsSystem](#), [MovementSystem](#), [TimeSystem](#), and [EventSystem](#).

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

- systems/System.hpp

## 6.95 TaskComponent Struct Reference

Defines a task by giving it a type, source (the task handler) and a target (subject of the task).

```
#include <Components.hpp>
```

### Public Member Functions

- **TaskComponent** (tdt::uint target\_id=Component::NO\_ENTITY, tdt::uint source\_id=Component::NO\_ENTITY, TASK\_TYPE t\_type=TASK\_TYPE::NONE)
- **TaskComponent** (const [TaskComponent](#) &)=default
- **TaskComponent** ([TaskComponent](#) &&)=default
- [TaskComponent](#) & **operator=** (const [TaskComponent](#) &)=default
- [TaskComponent](#) & **operator=** ([TaskComponent](#) &&)=default

### Public Attributes

- TASK\_TYPE **task\_type**
- tdt::uint **source**
- tdt::uint **target**
- bool **complete**

## Static Public Attributes

- static constexpr int **type** = 15

### 6.95.1 Detailed Description

Defines a task by giving it a type, source (the task handler) and a target (subject of the task).

Handling of these tasks is done via the [TaskHandlerComponent](#) below.

Definition at line 390 of file Components.hpp.

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

- Components.hpp

## 6.96 TaskHandlerComponent Struct Reference

Task queue and register of possible tasks, every entity that is able to actually do something on it's own should have it.

```
#include <Components.hpp>
```

## Public Member Functions

- **TaskHandlerComponent** (std::string &&b="ERROR")
- **TaskHandlerComponent** (const [TaskHandlerComponent](#) &)=default
- **TaskHandlerComponent** ([TaskHandlerComponent](#) &&)=default
- [TaskHandlerComponent](#) & **operator=** (const [TaskHandlerComponent](#) &)=default
- [TaskHandlerComponent](#) & **operator=** ([TaskHandlerComponent](#) &&)=default

## Public Attributes

- tdt::uint **curr\_task**
- std::bitset<(int) TASK\_TYPE::COUNT > **possible\_tasks**
- std::deque< tdt::uint > **task\_queue**
- bool **busy**
- std::string **blueprint**

## Static Public Attributes

- static constexpr int **type** = 16

### 6.96.1 Detailed Description

Task queue and register of possible tasks, every entity that is able to actually do something on it's own should have it.

Definition at line 415 of file Components.hpp.

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

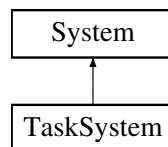
- Components.hpp

## 6.97 TaskSystem Class Reference

[System](#) managing all entities with the [TaskComponent](#), their creation, assignment, lifetime checks and canceling.

```
#include <TaskSystem.hpp>
```

Inheritance diagram for TaskSystem:



### Public Member Functions

- [TaskSystem](#) ([EntitySystem](#) &, [GridSystem](#) &, [CombatSystem](#) &)  
*Constructor.*
- [~TaskSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Manages the lifetime of tasks on each frame.*
- const std::string & [get\\_task\\_name](#) (TASK\_TYPE) const  
*Translates a task type enum value into a string that can be displayed in the developer's console.*

### Private Member Functions

- void [next\\_task\\_](#) ([TaskHandlerComponent](#) &)  
*Set's the current task to the task in the front of the respective task queue and removes it from the queue.*
- bool [handle\\_task\\_](#) (tdt::uint, [TaskHandlerComponent](#) &)  
*Executes a single task.*
- bool [current\\_task\\_completed\\_](#) (tdt::uint, [TaskHandlerComponent](#) &)  
*Checks whether the current task of a given entity has been completed.*



## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Reference to the game's entity system.*
- [GridSystem](#) & [grid\\_](#)  
*Reference to the game's grid system.*
- [CombatSystem](#) & [combat\\_](#)  
*Reference to the game's combat system used for line of sight checking.*
- `std::map< TASK_TYPE, std::string >` [task\\_names\\_](#)  
*Map used for task type translation.*

### 6.97.1 Detailed Description

[System](#) managing all entities with the [TaskComponent](#), their creation, assignment, lifetime checks and canceling.

Definition at line 18 of file TaskSystem.hpp.

### 6.97.2 Constructor & Destructor Documentation

#### 6.97.2.1 TaskSystem::TaskSystem ( EntitySystem & ents, GridSystem & grid, CombatSystem & comb )

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
<i>Reference</i>	to the game's grid system.

Definition at line 7 of file TaskSystem.cpp.

#### 6.97.2.2 TaskSystem::~~TaskSystem ( ) [default]

Destructor.

### 6.97.3 Member Function Documentation

#### 6.97.3.1 bool TaskSystem::current\_task\_completed\_ ( tdt::uint id, TaskHandlerComponent & handler ) [private]

Checks whether the current task of a given entity has been completed.

Parameters

<i>ID</i>	of the handling entity.
<i>Reference</i>	to the entity's <a href="#">TaskHandlerComponent</a> .

Definition at line 70 of file TaskSystem.cpp.

#### 6.97.3.2 `const std::string & TaskSystem::get_task_name ( TASK_TYPE type ) const`

Translates a task type enum value into a string that can be displayed in the developer's console.

##### Parameters

<i>Task</i>	type to be translated.
-------------	------------------------

Definition at line 45 of file TaskSystem.cpp.

#### 6.97.3.3 `bool TaskSystem::handle_task_ ( tdt::uint id, TaskHandlerComponent & handler ) [private]`

Executes a single task.

##### Parameters

<i>ID</i>	of the entity that the task is assigned to.
<i>Reference</i>	to the task handling component of the assigned entity. (For easier look up of the blueprint.)

Definition at line 63 of file TaskSystem.cpp.

#### 6.97.3.4 `void TaskSystem::next_task_ ( TaskHandlerComponent & comp ) [private]`

Set's the current task to the task in the front of the respective task queue and removes it from the queue.

##### Parameters

<i>Reference</i>	to the <a href="#">TaskHandlerComponent</a> containing the aforementioned task queue.
------------------	---

Definition at line 54 of file TaskSystem.cpp.

#### 6.97.3.5 `void TaskSystem::update ( tdt::real delta ) [override],[virtual]`

Manages the lifetime of tasks on each frame.

##### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 16 of file TaskSystem.cpp.

### 6.97.4 Member Data Documentation

#### 6.97.4.1 CombatSystem& TaskSystem::combat\_ [private]

Reference to the game's combat system used for line of sight checking.

Definition at line 82 of file TaskSystem.hpp.

#### 6.97.4.2 EntitySystem& TaskSystem::entities\_ [private]

Reference to the game's entity system.

Definition at line 72 of file TaskSystem.hpp.

#### 6.97.4.3 GridSystem& TaskSystem::grid\_ [private]

Reference to the game's grid system.

Definition at line 77 of file TaskSystem.hpp.

#### 6.97.4.4 std::map<TASK\_TYPE, std::string> TaskSystem::task\_names\_ [private]

Map used for task type translation.

Definition at line 87 of file TaskSystem.hpp.

The documentation for this class was generated from the following files:

- systems/TaskSystem.hpp
- systems/TaskSystem.cpp

## 6.98 TimeComponent Struct Reference

Represents a timer that after a certain amount of time can start end an event (it's target).

```
#include <Components.hpp>
```

### Public Member Functions

- **TimeComponent** (TIME\_EVENT ev=TIME\_EVENT::NONE, tdt::real limit=0.f, tdt::uint t=Component::NO\_ENTITY)
- **TimeComponent** (const [TimeComponent](#) &)=default
- **TimeComponent** ([TimeComponent](#) &&)=default
- **TimeComponent** & **operator=** (const [TimeComponent](#) &)=default
- **TimeComponent** & **operator=** ([TimeComponent](#) &&)=default

## Public Attributes

- tdt::real **curr\_time**
- tdt::real **time\_limit**
- tdt::uint **target**
- TIME\_EVENT **event\_type**

## Static Public Attributes

- static constexpr int **type** = 8

### 6.98.1 Detailed Description

Represents a timer that after a certain amount of time can start end an event (it's target).

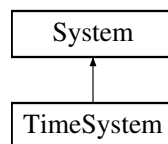
Definition at line 228 of file Components.hpp.

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

- Components.hpp

## 6.99 TimeSystem Class Reference

Inheritance diagram for TimeSystem:



## Public Member Functions

- [TimeSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~TimeSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Updates the time passed for all TimeComponents and handles those that surpassed their target time.*
- void [advance\\_all\\_timers](#) (tdt::real)  
*Adds a given time value to all TimeComponents.*
- void [advance\\_all\\_timers\\_of\\_type](#) (tdt::real, TIME\_EVENT)  
*Adds a given time value to all TimeComponents that match the given time event type.*
- void [set\\_time\\_multiplier](#) (tdt::real=1.f)  
*Sets the time value by which are all frame times multiplied when added to timers (this allows to slow/speed up all timers).*
- tdt::real [get\\_time\\_multiplier](#) ()  
*Returns the time value by which are all frame times multiplied when added to timers.*

## Private Member Functions

- void `handle_event_` (tdt::uint, `TimeComponent` &)  
*Handles a time event when it's timer finishes.*

## Private Attributes

- `EntitySystem` & `entities_`  
*Reference to the game's entity system.*
- tdt::real `time_multiplier_`  
*Allows to speed up all timers.*

### 6.99.1 Detailed Description

Definition at line 9 of file `TimeSystem.hpp`.

### 6.99.2 Constructor & Destructor Documentation

#### 6.99.2.1 `TimeSystem::TimeSystem ( EntitySystem & ents )`

Constructor.

Parameters

<i>Reference</i>	to the game's entity system.
------------------	------------------------------

Definition at line 7 of file `TimeSystem.cpp`.

#### 6.99.2.2 `TimeSystem::~~TimeSystem ( )` [default]

Destructor.

### 6.99.3 Member Function Documentation

#### 6.99.3.1 `void TimeSystem::advance_all_timers ( tdt::real delta )`

Adds a given time value to all `TimeComponents`.

Parameters

<i>Time</i>	to add.
-------------	---------

Definition at line 48 of file `TimeSystem.cpp`.

### 6.99.3.2 void TimeSystem::advance\_all\_timers\_of\_type ( tdt::real *delta*, TIME\_EVENT *type* )

Adds a given time value to all TimeComponents that match the given time event type.

#### Parameters

<i>Time</i>	to add.
<i>Time</i>	even type to match.

Definition at line 54 of file TimeSystem.cpp.

### 6.99.3.3 tdt::real TimeSystem::get\_time\_multiplier ( )

Returns the time value by which are all frame times multiplied when added to timers.

Definition at line 68 of file TimeSystem.cpp.

### 6.99.3.4 void TimeSystem::handle\_event\_ ( tdt::uint *id*, TimeComponent & *comp* ) [private]

Handles a time event when it's timer finishes.

#### Parameters

<i>ID</i>	of the time event.
<i>Reference</i>	to the <a href="#">TimeComponent</a> of this time event.

Definition at line 73 of file TimeSystem.cpp.

### 6.99.3.5 void TimeSystem::set\_time\_multiplier ( tdt::real *val* = 1.f )

Sets the time value by which are all frame times multiplied when added to timers (this allows to slow/speed up all timers).

#### Parameters

<i>The</i>	new time multiplier.
------------	----------------------

Definition at line 63 of file TimeSystem.cpp.

### 6.99.3.6 void TimeSystem::update ( tdt::real *delta* ) [override],[virtual]

Updates the time passed for all TimeComponents and handles those that surpassed their target time.

#### Parameters

<i>Time</i>	since last frame.
-------------	-------------------

Implements [System](#).

Definition at line 11 of file TimeSystem.cpp.

## 6.99.4 Member Data Documentation

### 6.99.4.1 EntitySystem& TimeSystem::entities\_ [private]

Reference to the game's entity system.

Definition at line 68 of file TimeSystem.hpp.

### 6.99.4.2 tdt::real TimeSystem::time\_multiplier\_ [private]

Allows to speed up all timers.

Definition at line 73 of file TimeSystem.hpp.

The documentation for this class was generated from the following files:

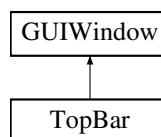
- systems/TimeSystem.hpp
- systems/TimeSystem.cpp

## 6.100 TopBar Class Reference

Class representing an info bar on the top of the screen displaying the name of the game, player's gold, mana, units and the current time.

```
#include <TopBar.hpp>
```

Inheritance diagram for TopBar:



## Public Member Functions

- [TopBar](#) ()  
*Constructor.*
- [~TopBar](#) ()=default  
*Destructor.*
- void [update\\_time](#) (tdt::real)  
*Updates the current time on the top bar if a second passed since the last time update.*
- void [update\\_label](#) (const std::string &, const std::string &)  
*Sets the given label's text to the given string.*

## Protected Member Functions

- void `init_()` override  
*Initializes the top bar.*

## Private Attributes

- `tdt::real` `tdelta_`  
*Time since the last "Current Time" update.*

## Additional Inherited Members

### 6.100.1 Detailed Description

Class representing an info bar on the top of the screen displaying the name of the game, player's gold, mana, units and the current time.

Definition at line 12 of file `TopBar.hpp`.

### 6.100.2 Constructor & Destructor Documentation

#### 6.100.2.1 `TopBar::TopBar ( )`

Constructor.

Definition at line 5 of file `TopBar.cpp`.

#### 6.100.2.2 `TopBar::~~TopBar ( )` [default]

Destructor.

### 6.100.3 Member Function Documentation

#### 6.100.3.1 `void TopBar::init_ ( )` [override], [protected], [virtual]

Initializes the top bar.

Implements [GUIWindow](#).

Definition at line 28 of file `TopBar.cpp`.

#### 6.100.3.2 `void TopBar::update_label ( const std::string & label, const std::string & val )`

Sets the given label's text to the given string.



## Parameters

<i>Label</i>	to change.
<i>New</i>	text.

Definition at line 23 of file TopBar.cpp.

6.100.3.3 void TopBar::update\_time ( tdt::real *delta* )

Updates the current time on the top bar if a second passed since the last time update.

## Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Definition at line 9 of file TopBar.cpp.

## 6.100.4 Member Data Documentation

## 6.100.4.1 tdt::real TopBar::tdelta\_ [private]

Time since the last "Current Time" update.

Definition at line 49 of file TopBar.hpp.

The documentation for this class was generated from the following files:

- gui/TopBar.hpp
- gui/TopBar.cpp

## 6.101 TriggerComponent Struct Reference

Allows an entity to cause an effect (by calling it's blueprint) when its triggered (stepped on) or can notify a linked entity which causes the effect.

```
#include <Components.hpp>
```

## Public Member Functions

- **TriggerComponent** (std::string &&b="ERROR", tdt::real cd=0.f, tdt::real rad=0.f)
- **TriggerComponent** (const [TriggerComponent](#) &)=default
- **TriggerComponent** ([TriggerComponent](#) &&)=default
- [TriggerComponent](#) & **operator=** (const [TriggerComponent](#) &)=default
- [TriggerComponent](#) & **operator=** ([TriggerComponent](#) &&)=default

## Public Attributes

- `std::string` **blueprint**
- `tdt::uint` **linked\_entity**
- `tdt::real` **curr\_time**
- `tdt::real` **cooldown**
- `tdt::real` **radius**

## Static Public Attributes

- `static constexpr int` **type** = 29

### 6.101.1 Detailed Description

Allows an entity to cause an effect (by calling it's blueprint) when its triggered (stepped on) or can notify a linked entity which causes the effect.

Definition at line 689 of file Components.hpp.

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

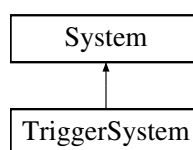
- Components.hpp

## 6.102 TriggerSystem Class Reference

Handles triggers by checking if an entity is standing in their radius when they are off cooldowns.

```
#include <TriggerSystem.hpp>
```

Inheritance diagram for TriggerSystem:



## Public Member Functions

- `TriggerSystem (EntitySystem &)`  
*Constructor.*
- `~TriggerSystem ()=default`  
*Destructor.*
- `void update (tdt::real)` override  
*Checks if any entities have been triggered and performs their associated actions if they were.*
- `void set_check_period (tdt::real)`  
*Sets the time period between trigger checks.*
- `tdt::real get_check_period () const`  
*Returns the time period between trigger checks.*

## Private Attributes

- [EntitySystem](#) & [entities\\_](#)  
*Entity system containing the entities this system works with.*
- tdt::real [check\\_timer\\_](#)  
*Allow for dynamic time periods between trigger checks.*
- tdt::real [check\\_period\\_](#)

### 6.102.1 Detailed Description

Handles triggers by checking if an entity is standing in their radius when they are off cooldowns.

Definition at line 11 of file TriggerSystem.hpp.

### 6.102.2 Constructor & Destructor Documentation

#### 6.102.2.1 TriggerSystem::TriggerSystem ( EntitySystem & ents )

Constructor.

Parameters

<i>Entity</i>	system containing entities this system works with.
---------------	--

Definition at line 6 of file TriggerSystem.cpp.

#### 6.102.2.2 TriggerSystem::~~TriggerSystem ( ) [default]

Destructor.

### 6.102.3 Member Function Documentation

#### 6.102.3.1 tdt::real TriggerSystem::get\_check\_period ( ) const

Returns the time period between trigger checks.

Definition at line 65 of file TriggerSystem.cpp.

#### 6.102.3.2 void TriggerSystem::set\_check\_period ( tdt::real val )

Sets the time period between trigger checks.

Parameters

<i>The</i>	new time period.
------------	------------------

Definition at line 60 of file TriggerSystem.cpp.

**6.102.3.3** `void TriggerSystem::update ( tdt::real delta )` `[override], [virtual]`

Checks if any entities have been triggered and performs their associated actions if they were.

#### Parameters

<i>Time</i>	since the last frame.
-------------	-----------------------

Implements [System](#).

Definition at line 10 of file TriggerSystem.cpp.

## 6.102.4 Member Data Documentation

**6.102.4.1** `tdt::real TriggerSystem::check_timer_` `[private]`

Allow for dynamic time periods between trigger checks.

Definition at line 53 of file TriggerSystem.hpp.

**6.102.4.2** `EntitySystem& TriggerSystem::entities_` `[private]`

Entity system containing the entities this system works with.

Definition at line 48 of file TriggerSystem.hpp.

The documentation for this class was generated from the following files:

- systems/TriggerSystem.hpp
- systems/TriggerSystem.cpp

## 6.103 UpgradeComponent Struct Reference

Represents the game's leveling system component, contains info about experience and leveling progression as well as the blueprint that gets called on level up.

```
#include <Components.hpp>
```

### Public Member Functions

- **UpgradeComponent** (std::string &&b="ERROR", tdt::uint exp=100, tdt::uint cap=5)
- **UpgradeComponent** (const [UpgradeComponent](#) &)=default
- **UpgradeComponent** ([UpgradeComponent](#) &&)=default
- **UpgradeComponent & operator=** (const [UpgradeComponent](#) &)=default
- **UpgradeComponent & operator=** ([UpgradeComponent](#) &&)=default

## Public Attributes

- `std::string` **blueprint**
- `tdt::uint` **experience**
- `tdt::uint` **exp\_needed**
- `tdt::uint` **level**
- `tdt::uint` **level\_cap**

## Static Public Attributes

- static constexpr int **type** = 30

### 6.103.1 Detailed Description

Represents the game's leveling system component, contains info about experience and leveling progression as well as the blueprint that gets called on level up.

Definition at line 714 of file Components.hpp.

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

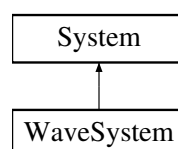
- Components.hpp

## 6.104 WaveSystem Class Reference

This system creates the entities attacking the player's dungeon in a similar fashion to tower defense games.

```
#include <WaveSystem.hpp>
```

Inheritance diagram for WaveSystem:



## Public Member Functions

- [WaveSystem](#) ([EntitySystem](#) &)  
*Constructor.*
- [~WaveSystem](#) ()=default  
*Destructor.*
- void [update](#) (tdt::real) override  
*Updates all necessary timers and checks if a wave has arrived and if so, spawns it.*
- void [set\\_countdown\\_window](#) (CEGUI::Window \*)  
*Saves pointer to the window that is used to show the countdown to the next wave.*
- void [next\\_wave](#) ()  
*Finishes the wave countdown, causing a wave to be spawned on the next update call.*
- void [advance\\_wave\\_countdown](#) (tdt::uint)  
*Advances the countdown by a given amount of time.*
- void [wave\\_entity\\_died](#) ()  
*Registers a death of an entity that belongs to the current wave (called by it's destructor).*
- void [start](#) ()  
*Starts (and initializes) the wave system.*
- void [pause](#) (bool)  
*Changes the state of the system.*
- void [set\\_entity\\_total](#) (tdt::uint)  
*Sets the total number of entities the wave is gonna have.*
- tdt::uint [get\\_entity\\_total](#) () const  
*Returns the total number of entities the wave is gonna have.*
- void [set\\_wave\\_count](#) (tdt::uint)  
*Sets the number of waves before the system stops.*
- tdt::uint [get\\_wave\\_count](#) () const  
*Returns the total number of waves the system has.*
- void [add\\_spawn\\_node](#) (tdt::uint)  
*Adds a grid node to the spawn list.*
- void [clear\\_spawn\\_nodes](#) ()  
*Removes any registered spawn nodes.*
- void [set\\_spawn\\_cooldown](#) (tdt::real)  
*Sets the delay between spawns (if # of spawn nodes is smaller than # of wave entities).*
- tdt::real [get\\_spawn\\_cooldown](#) () const  
*Returns the delay between spawns.*
- void [update\\_label\\_text](#) ()  
*Updates the countdown window's text.*
- void [add\\_entity\\_blueprint](#) (const std::string &)  
*Adds a blueprint table for a new wave entity.*
- const std::vector< std::string > & [get\\_entity\\_blueprints](#) () const  
*Returns a vector of all entities in the wave.*
- void [set\\_wave\\_table](#) (const std::string &)  
*Sets the table this system is using to create waves.*
- const std::string & [get\\_wave\\_table](#) () const  
*Returns the table this system is using to create waves.*
- const std::vector< tdt::uint > & [get\\_spawning\\_nodes](#) () const  
*Returns a vector of nodes that are marked as spawners for the wave.*
- void [set\\_curr\\_wave\\_number](#) (tdt::uint)  
*Sets the number of the current wave.*
- tdt::uint [get\\_curr\\_wave\\_number](#) () const

- Returns the number of the current wave.*
- void [set\\_countdown\\_value](#) (tdt::uint)
  - Sets the time remaining before the next wave.*
- tdt::uint [get\\_countdown\\_value](#) () const
  - Returns the time remaining before the next wave.*
- void [set\\_state](#) (WAVE\_STATE)
  - Changes the state of the wave system.*
- WAVE\_STATE [get\\_state](#) () const
  - Returns the state of the wave system.*
- void [set\\_spawn\\_timer](#) (tdt::real)
  - Sets the time on the spawn timer, which causes the next spawn batch to appear when it reaches the spawn cooldown value.*
- tdt::real [get\\_spawn\\_timer](#) () const
  - Returns the time on the spawn timer, which causes the next spawn batch to appear when it reaches the spawn cooldown value.*
- void [set\\_wave\\_entities](#) (tdt::uint)
  - Sets the number of entities the current wave has alive.*
- tdt::uint [get\\_wave\\_entities](#) () const
  - Returns the number of entities the current wave has alive.*
- void [set\\_entities\\_spawned](#) (tdt::uint)
  - Sets the number of entities already spawned in this wave.*
- tdt::uint [get\\_entities\\_spawned](#) () const
  - Returns the number of entities already spawned in this wave.*
- void [clear\\_entity\\_blueprints](#) ()
  - Clears any entity blueprints that were gonna be used in the next wave.*
- void [set\\_endless\\_mode](#) (bool)
  - Sets the endless flag (true -> last wave repeats).*
- bool [get\\_endless\\_mode](#) () const
  - Returns the endless flag (true -> last wave repeats).*

### Private Member Functions

- void [start\\_wave](#) ()
  - Starts the current wave.*
- void [end\\_wave](#) ()
  - Ends the current wave.*
- void [spawn](#) ()
  - Spawns the next batch of entities.*

### Private Attributes

- WAVE\_STATE [state\\_](#)
  - The current state of the system.*
- EntitySystem & [entities\\_](#)
  - Entity system in which the wave entities will be created.*
- tdt::uint [curr\\_wave\\_number\\_](#)
  - Number of the current wave.*
- tdt::uint [wave\\_count\\_](#)
  - Total number of waves.*
- tdt::uint [wave\\_entities\\_](#)

- *Number of entities in the current wave that are spawned and still alive.*  
tdt::uint [entities\\_spawned\\_](#)
- *Number of entities that were already spawned during this wave.*  
tdt::uint [entities\\_total\\_](#)
- *Total number of entities in the current wave.*  
tdt::uint [next\\_wave\\_countdown\\_](#)
- *Time in seconds till the next wave starts.*  
tdt::real [second\\_timer\\_](#)
- *Auxiliary timer that is used to measure seconds.*  
std::string [label\\_text\\_](#)
- *Text that is displayed in the countdown window.*  
CEGUI::Window \* [window\\_](#)
- *Pointer to the countdown window.*  
[lpp::Script](#) & [script\\_](#)
- *Reference to the script engine for easier use.*  
std::vector< tdt::uint > [spawning\\_nodes\\_](#)
- *Nodes that are used for entity spawning.*  
tdt::real [spawn\\_timer\\_](#)
- *Timers used for entity spawning.*  
tdt::real [spawn\\_cooldown\\_](#)
- *Name of the table containing init, wstart and wend functions that define a wave system.*  
std::string [wave\\_table\\_](#)
- *Entities that are gonna be spawned.*  
std::vector< std::string > [entity\\_blueprints\\_](#)
- *If true, the current wave keeps repeating.*  
bool [endless\\_mode\\_](#)

### 6.104.1 Detailed Description

This system creates the entities attacking the player's dungeon in a similar fashion to tower defense games.

#### Note

This systems contains a large number of setters/getters (while most of other systems do not). This is because this system has to be fully serialized when saving the game to save the game's progress.

Definition at line 25 of file WaveSystem.hpp.

### 6.104.2 Constructor & Destructor Documentation

#### 6.104.2.1 WaveSystem::WaveSystem ( EntitySystem & ents )

Constructor.

#### Parameters

<a href="#">EntitySystem</a>	in which wave entities will be created.
------------------------------	---



Definition at line 7 of file WaveSystem.cpp.

#### 6.104.2.2 WaveSystem::~WaveSystem ( ) [default]

Destructor.

### 6.104.3 Member Function Documentation

#### 6.104.3.1 void WaveSystem::add\_entity\_blueprint ( const std::string & val )

Adds a blueprint table for a new wave entity.

Parameters

<i>Name</i>	of the blueprint table.
-------------	-------------------------

Definition at line 169 of file WaveSystem.cpp.

#### 6.104.3.2 void WaveSystem::add\_spawn\_node ( tdt::uint id )

Adds a grid node to the spawn list.

Parameters

<i>ID</i>	of the node.
-----------	--------------

Definition at line 117 of file WaveSystem.cpp.

#### 6.104.3.3 void WaveSystem::advance\_wave\_countdown ( tdt::uint val )

Advances the countdown by a given amount of time.

Parameters

<i>Number</i>	of seconds to subtract from the countdown.
---------------	--

Definition at line 64 of file WaveSystem.cpp.

#### 6.104.3.4 void WaveSystem::clear\_entity\_blueprints ( )

Clears any entity blueprints that were gonna be used in the next wave.

Definition at line 254 of file WaveSystem.cpp.

#### 6.104.3.5 void WaveSystem::clear\_spawn\_nodes ( )

Removes any registered spawn nodes.

Definition at line 122 of file WaveSystem.cpp.

#### 6.104.3.6 void WaveSystem::end\_wave ( ) [private]

Ends the current wave.

Definition at line 280 of file WaveSystem.cpp.

#### 6.104.3.7 tdt::uint WaveSystem::get\_countdown\_value ( ) const

Returns the time remaining before the next wave.

Definition at line 209 of file WaveSystem.cpp.

#### 6.104.3.8 tdt::uint WaveSystem::get\_curr\_wave\_number ( ) const

Returns the number of the current wave.

Definition at line 199 of file WaveSystem.cpp.

#### 6.104.3.9 bool WaveSystem::get\_endless\_mode ( ) const

Returns the endless flag (true -> last wave repeats).

Definition at line 264 of file WaveSystem.cpp.

#### 6.104.3.10 tdt::uint WaveSystem::get\_entities\_spawned ( ) const

Returns the number of entities already spawned in this wave.

Definition at line 249 of file WaveSystem.cpp.

#### 6.104.3.11 const std::vector< std::string > & WaveSystem::get\_entity\_blueprints ( ) const

Returns a vector of all entities in the wave.

Definition at line 174 of file WaveSystem.cpp.

#### 6.104.3.12 tdt::uint WaveSystem::get\_entity\_total ( ) const

Returns the total number of entities the wave is gonna have.

Definition at line 102 of file WaveSystem.cpp.

**6.104.3.13** `tdt::real WaveSystem::get_spawn_cooldown ( ) const`

Returns the delay between spawns.

Definition at line 132 of file WaveSystem.cpp.

**6.104.3.14** `tdt::real WaveSystem::get_spawn_timer ( ) const`

Returns the time on the spawn timer, which causes the next spawn batch to appear when it reaches the spawn cooldown value.

Definition at line 229 of file WaveSystem.cpp.

**6.104.3.15** `const std::vector< tdt::uint > & WaveSystem::get_spawning_nodes ( ) const`

Returns a vector of nodes that are marked as spawners for the wave.

Definition at line 189 of file WaveSystem.cpp.

**6.104.3.16** `WAVE_STATE WaveSystem::get_state ( ) const`

Returns the state of the wave system.

Definition at line 219 of file WaveSystem.cpp.

**6.104.3.17** `tdt::uint WaveSystem::get_wave_count ( ) const`

Returns the total number of waves the system has.

Definition at line 112 of file WaveSystem.cpp.

**6.104.3.18** `tdt::uint WaveSystem::get_wave_entities ( ) const`

Returns the number of entities the current wave has alive.

Definition at line 239 of file WaveSystem.cpp.

**6.104.3.19** `const std::string & WaveSystem::get_wave_table ( ) const`

Returns the table this system is using to create waves.

Definition at line 184 of file WaveSystem.cpp.

**6.104.3.20** `void WaveSystem::next_wave ( )`

Finnishes the wave countdown, causing a wave to be spawned on the next update call.

Definition at line 59 of file WaveSystem.cpp.

**6.104.3.21** `void WaveSystem::pause ( bool val )`

Changes the state of the system.

## Parameters

<i>If</i>	true, the wave countdown gets paused, if false, the wave countdown gets resumed.
-----------	--

Definition at line 89 of file WaveSystem.cpp.

6.104.3.22 void WaveSystem::set\_countdown\_value ( tdt::uint *val* )

Sets the time remaining before the next wave.

## Parameters

<i>The</i>	new countdown time value.
------------	---------------------------

Definition at line 204 of file WaveSystem.cpp.

6.104.3.23 void WaveSystem::set\_countdown\_window ( CEGUI::Window \* *win* )

Saves pointer to the window that is used to show the countdown to the next wave.

## Parameters

<i>The</i>	new window.
------------	-------------

Definition at line 52 of file WaveSystem.cpp.

6.104.3.24 void WaveSystem::set\_curr\_wave\_number ( tdt::uint *val* )

Sets the number of the current wave.

## Parameters

<i>The</i>	new wave number.
------------	------------------

Definition at line 194 of file WaveSystem.cpp.

6.104.3.25 void WaveSystem::set\_endless\_mode ( bool *val* )

Sets the endless flag (true -> last wave repeats).

## Parameters

<i>If</i>	true, endless mode is turned on, otherwise it's turned off.
-----------	---

Definition at line 259 of file WaveSystem.cpp.

6.104.3.26 void WaveSystem::set\_entities\_spawned ( tdt::uint val )

Sets the number of entities already spawned in this wave.

#### Parameters

<i>The</i>	new entity count.
------------	-------------------

Definition at line 244 of file WaveSystem.cpp.

6.104.3.27 void WaveSystem::set\_entity\_total ( tdt::uint val )

Sets the total number of entities the wave is gonna have.

#### Parameters

<i>The</i>	new number of entities.
------------	-------------------------

Definition at line 97 of file WaveSystem.cpp.

6.104.3.28 void WaveSystem::set\_spawn\_cooldown ( tdt::real val )

Sets the delay between spawns (if # of spawn nodes is smaller than # of wave entities).

#### Parameters

<i>The</i>	new spawn cooldown.
------------	---------------------

Definition at line 127 of file WaveSystem.cpp.

6.104.3.29 void WaveSystem::set\_spawn\_timer ( tdt::real val )

Sets the time on the spawn timer, which causes the next spawn batch to appear when it reaches the spawn cooldown value.

#### Parameters

<i>The</i>	new spawn timer value.
------------	------------------------

Definition at line 224 of file WaveSystem.cpp.

#### 6.104.3.30 void WaveSystem::set\_state ( WAVE\_STATE *val* )

Changes the state of the wave system.

##### Parameters

<i>The</i>	new state.
------------	------------

Definition at line 214 of file WaveSystem.cpp.

#### 6.104.3.31 void WaveSystem::set\_wave\_count ( tdt::uint *val* )

Sets the number of waves before the system stops.

##### Parameters

<i>The</i>	new number of waves.
------------	----------------------

Definition at line 107 of file WaveSystem.cpp.

#### 6.104.3.32 void WaveSystem::set\_wave\_entities ( tdt::uint *val* )

Sets the number of entities the current wave has alive.

##### Parameters

<i>The</i>	enw number of entities.
------------	-------------------------

Definition at line 234 of file WaveSystem.cpp.

#### 6.104.3.33 void WaveSystem::set\_wave\_table ( const std::string & *val* )

Sets the table this system is using to create waves.

##### Parameters

<i>Name</i>	of the new wave table.
-------------	------------------------

Definition at line 179 of file WaveSystem.cpp.

#### 6.104.3.34 void WaveSystem::spawn ( ) [private]

Spawns the next batch of entities.

Definition at line 293 of file WaveSystem.cpp.

**6.104.3.35** void WaveSystem::start ( )

Starts (and initializes) the wave system.

Definition at line 81 of file WaveSystem.cpp.

**6.104.3.36** void WaveSystem::start\_wave ( ) [private]

Starts the current wave.

Definition at line 269 of file WaveSystem.cpp.

**6.104.3.37** void WaveSystem::update ( tdt::real *delta* ) [override],[virtual]

Updates all necessary timers and checks if a wave has arrived and if so, spawns it.

**Parameters**

<i>Time</i>	since last frame.
-------------	-------------------

Implements [System](#).

Definition at line 15 of file WaveSystem.cpp.

**6.104.3.38** void WaveSystem::update\_label\_text ( )

Updates the countdown window's text.

Definition at line 137 of file WaveSystem.cpp.

**6.104.3.39** void WaveSystem::wave\_entity\_died ( )

Registers a death of an entity that belongs to the current wave (called by it's destructor).

Definition at line 72 of file WaveSystem.cpp.

**6.104.4 Member Data Documentation****6.104.4.1** tdt::uint WaveSystem::curr\_wave\_number\_ [private]

Number of the current wave.

Definition at line 278 of file WaveSystem.hpp.

#### 6.104.4.2 `bool WaveSystem::endless_mode_` [private]

If true, the current wave keeps repeating.

Definition at line 351 of file WaveSystem.hpp.

#### 6.104.4.3 `EntitySystem& WaveSystem::entities_` [private]

Entity system in which the wave entities will be created.

Definition at line 273 of file WaveSystem.hpp.

#### 6.104.4.4 `tdt::uint WaveSystem::entities_spawned_` [private]

Number of entities that were already spawned during this wave.

Definition at line 295 of file WaveSystem.hpp.

#### 6.104.4.5 `tdt::uint WaveSystem::entities_total_` [private]

Total number of entities in the current wave.

Definition at line 300 of file WaveSystem.hpp.

#### 6.104.4.6 `std::vector<std::string> WaveSystem::entity_blueprints_` [private]

Entities that are gonna be spawned.

Definition at line 346 of file WaveSystem.hpp.

#### 6.104.4.7 `std::string WaveSystem::label_text_` [private]

Text that is displayed in the countdown window.

Definition at line 315 of file WaveSystem.hpp.

#### 6.104.4.8 `tdt::uint WaveSystem::next_wave_countdown_` [private]

Time in seconds till the next wave starts.

Definition at line 305 of file WaveSystem.hpp.

#### 6.104.4.9 `lpp::Script& WaveSystem::script_` [private]

Reference to the script engine for easier use.

Definition at line 325 of file WaveSystem.hpp.



**6.104.4.10** `tdt::real WaveSystem::second_timer_ [private]`

Auxiliary timer that is used to measure seconds.

Definition at line 310 of file WaveSystem.hpp.

**6.104.4.11** `tdt::real WaveSystem::spawn_timer_ [private]`

Timers used for entity spawning.

Definition at line 335 of file WaveSystem.hpp.

**6.104.4.12** `std::vector<tdt::uint> WaveSystem::spawning_nodes_ [private]`

Nodes that are used for entity spawning.

Definition at line 330 of file WaveSystem.hpp.

**6.104.4.13** `WAVE_STATE WaveSystem::state_ [private]`

The current state of the system.

Definition at line 268 of file WaveSystem.hpp.

**6.104.4.14** `tdt::uint WaveSystem::wave_count_ [private]`

Total number of waves.

Definition at line 283 of file WaveSystem.hpp.

**6.104.4.15** `tdt::uint WaveSystem::wave_entities_ [private]`

Number of entities in the current wave that are spawned and still alive.

Definition at line 289 of file WaveSystem.hpp.

**6.104.4.16** `std::string WaveSystem::wave_table_ [private]`

Name of the table containing init, wstart and wend functions that define a wave system.

Definition at line 341 of file WaveSystem.hpp.

**6.104.4.17** `CEGUI::Window* WaveSystem::window_ [private]`

Pointer to the countdown window.

Definition at line 320 of file WaveSystem.hpp.

The documentation for this class was generated from the following files:

- systems/WaveSystem.hpp
- systems/WaveSystem.cpp



# Index

- ~AISystem
  - AISystem, [119](#)
- ~BuilderWindow
  - BuilderWindow, [124](#)
- ~Camera
  - Camera, [128](#)
- ~CombatSystem
  - CombatSystem, [136](#)
- ~Console
  - Console, [146](#)
- ~DAMAGE\_EFFECT
  - util::effect::DAMAGE\_EFFECT, [151](#)
- ~EntityCreator
  - EntityCreator, [154](#)
- ~EntityPlacer
  - EntityPlacer, [159](#)
- ~EntitySystem
  - EntitySystem, [169](#)
- ~EntityTracker
  - EntityTracker, [178](#)
- ~EventSystem
  - EventSystem, [184](#)
- ~FREEZE\_EFFECT
  - util::effect::FREEZE\_EFFECT, [191](#)
- ~GUI
  - GUI, [227](#)
- ~GUIWindow
  - GUIWindow, [235](#)
- ~Game
  - Game, [194](#)
- ~GameLog
  - GameLog, [202](#)
- ~GameSerializer
  - GameSerializer, [207](#)
- ~GraphicsSystem
  - GraphicsSystem, [212](#)
- ~Grid
  - Grid, [215](#)
- ~GridSystem
  - GridSystem, [223](#)
- ~HAS\_GOLD
  - util::HAS\_GOLD, [237](#)
- ~HEAL\_EFFECT
  - util::effect::HEAL\_EFFECT, [239](#)
- ~HealthSystem
  - HealthSystem, [241](#)
- ~IS\_ENEMY
  - util::IS\_ENEMY, [250](#)
- ~IS\_FRIENDLY
  - util::IS\_FRIENDLY, [252](#)
- ~IS\_FRIENDLY\_OR\_NEUTRAL
  - util::IS\_FRIENDLY\_OR\_NEUTRAL, [254](#)
- ~IS\_GOLD\_VAULT
  - util::IS\_GOLD\_VAULT, [255](#)
- ~InputSystem
  - InputSystem, [247](#)
- ~LOWER\_SPEED\_EFFECT
  - util::effect::LOWER\_SPEED\_EFFECT, [260](#)
- ~LevelGenerator
  - level\_generators::LevelGenerator, [257](#)
- ~ManaSpellSystem
  - ManaSpellSystem, [274](#)
- ~MessageToPlayerWindow
  - MessageToPlayerWindow, [277](#)
- ~MovementSystem
  - MovementSystem, [281](#)
- ~OptionsWindow
  - OptionsWindow, [287](#)
- ~Player
  - Player, [294](#)
- ~ProductionSystem
  - ProductionSystem, [306](#)
- ~ResearchWindow
  - ResearchWindow, [314](#)
- ~Script
  - lpp::Script, [320](#)
- ~SelectionBox
  - SelectionBox, [327](#)
- ~SpellCastingWindow
  - SpellCastingWindow, [338](#)
- ~Spellcaster
  - Spellcaster, [333](#)
- ~System
  - System, [344](#)
- ~TaskSystem
  - TaskSystem, [347](#)
- ~TimeSystem
  - TimeSystem, [351](#)
- ~TopBar
  - TopBar, [354](#)
- ~TriggerSystem
  - TriggerSystem, [357](#)
- ~WaveSystem
  - WaveSystem, [363](#)
- AIComponent, [118](#)
- AIHelper, [19](#)
  - get\_blueprint, [19](#)
  - get\_state, [20](#)

- set\_blueprint, [20](#)
  - set\_state, [20](#)
- AISystem, [118](#)
  - ~AISystem, [119](#)
  - AISystem, [119](#)
  - entities\_, [120](#)
  - force\_update, [120](#)
  - get\_update\_period, [120](#)
  - set\_update\_period, [120](#)
  - update, [120](#)
  - update\_timer\_, [120](#)
- abs
  - util, [113](#)
- action, [17](#)
  - BUILD\_TAB, [18](#)
  - CAST\_SPELL\_1, [18](#)
  - CAST\_SPELL\_2, [18](#)
  - CAST\_SPELL\_3, [18](#)
  - CAST\_SPELL\_4, [18](#)
  - MENU\_TAB, [18](#)
  - NEXT, [18](#)
  - PREV, [18](#)
  - QUICK\_LOAD, [19](#)
  - QUICK\_SAVE, [19](#)
  - RESET\_CAMERA, [19](#)
  - SPELL\_TAB, [19](#)
- actions\_
  - OptionsWindow, [289](#)
- actualize\_list
  - EntityCreator, [154](#)
- add\_component
  - EntitySystem, [169](#)
- add\_curr\_unit
  - Player, [295](#)
- add\_defense
  - HealthHelper, [55](#)
- add\_entity\_blueprint
  - WaveSystem, [363](#)
- add\_experience
  - UpgradeHelper, [109](#)
- add\_freed
  - Grid, [215](#)
- add\_gold
  - GoldHelper, [42](#)
  - Player, [295](#)
- add\_health
  - HealthHelper, [56](#)
- add\_mana
  - ManaHelper, [69](#)
  - Player, [295](#)
- add\_mana\_regen
  - Player, [295](#)
- add\_max\_mana
  - Player, [296](#)
- add\_max\_unit
  - Player, [296](#)
- add\_node
  - Grid, [215](#)
- add\_possible\_event
  - EventHandlerHelper, [32](#)
- add\_possible\_task
  - TaskHandlerHelper, [95](#)
- add\_residence
  - StructureHelper, [93](#)
- add\_residences
  - StructureHelper, [93](#)
- add\_spawn\_node
  - WaveSystem, [363](#)
- add\_start\_parameters
  - OptionsWindow, [288](#)
- add\_task
  - TaskHelper, [98](#)
- add\_unfreed
  - Grid, [216](#)
- advance\_all\_timers
  - TimeSystem, [351](#)
- advance\_all\_timers\_of\_type
  - TimeSystem, [351](#)
- advance\_curr\_time
  - LimitedLifeSpanHelper, [65](#)
  - NotificationHelper, [75](#)
  - SpellHelper, [90](#)
  - TimeHelper, [102](#)
- advance\_wave\_countdown
  - WaveSystem, [363](#)
- ai\_backup\_
  - InputSystem, [248](#)
- AlignComponent, [121](#)
- AlignComponent::AlignState, [122](#)
- apply\_
  - OptionsWindow, [288](#)
- apply\_damage\_to\_entities\_in\_range
  - CombatSystem, [136](#)
- apply\_effect\_to\_entities\_in\_range
  - CombatSystem, [137](#)
- apply\_freeze\_to
  - CombatSystem, [137](#)
- apply\_freeze\_to\_entities\_in\_range
  - CombatSystem, [137](#)
- apply\_heal\_to\_entities\_in\_range
  - CombatSystem, [137](#)
- apply\_scale
  - GraphicsHelper, [46](#)
- apply\_slow\_to
  - CombatSystem, [138](#)
- apply\_slow\_to\_entities\_in\_range
  - CombatSystem, [138](#)
- area\_free
  - GridNodeHelper, [51](#)
- BUILD\_TAB
  - action, [18](#)
- buff
  - HealthHelper, [56](#)
- build
  - BuilderWindow, [124](#)
- builder\_

- GUI, 232
- BuilderWindow, 122
  - ~BuilderWindow, 124
  - build, 124
  - BuilderWindow, 124
  - buildings\_, 126
  - clear\_buildings, 124
  - dec\_selection, 124
  - get\_building\_, 124
  - get\_buildings, 125
  - inc\_selection, 125
  - init\_, 125
  - placer\_, 126
  - register\_building, 125
  - selection\_number\_, 126
  - set\_placer, 125
  - update\_selection\_, 125
- buildings\_
  - BuilderWindow, 126
- CAST\_SPELL\_1
  - action, 18
- CAST\_SPELL\_2
  - action, 18
- CAST\_SPELL\_3
  - action, 18
- CAST\_SPELL\_4
  - action, 18
- call
  - ConstructorHelper, 28
  - lpp::Script, 320, 321
  - OnHitHelper, 77
- cam\_
  - InputSystem, 248
- cam\_position\_
  - InputSystem, 248
- Camera, 126
  - ~Camera, 128
  - Camera, 128
  - camera\_, 132
  - free\_mode\_, 132
  - get\_direction, 128
  - get\_free\_mode, 128
  - get\_height, 128
  - get\_position, 129
  - height\_, 132
  - init, 129
  - key\_pressed, 129
  - key\_released, 129
  - look\_at, 129
  - move, 130
  - movement\_direction\_, 132
  - pitch, 130
  - reset, 130
  - set\_direction, 130
  - set\_free\_mode, 130
  - set\_height, 131
  - set\_position, 131
  - set\_start, 131
  - speed\_, 132
  - start\_, 133
  - update, 131
  - yaw, 132
- camera\_
  - Camera, 132
- can\_be\_triggered\_by
  - TriggerHelper, 105
- can\_break
  - PathfindingHelper, 79
- can\_handle
  - EventHandlerHelper, 32
- can\_level\_up
  - UpgradeHelper, 109
- can\_move\_to
  - MovementSystem, 281
- can\_place\_when\_game\_paused
  - EntityPlacer, 159
- can\_place\_when\_game\_paused\_
  - EntityPlacer, 160
- cancel\_task
  - TaskHelper, 98
- cast
  - RayCaster, 311
  - SpellCastingWindow, 338
  - SpellHelper, 91
  - Spellcaster, 333
- caster\_
  - SpellCastingWindow, 340
- change\_to\_create
  - EntityCreator, 155
- change\_to\_place
  - EntityCreator, 155
- check\_timer\_
  - TriggerSystem, 358
- checked\_move
  - MovementSystem, 281
- clean\_up\_component
  - EntitySystem, 170
- cleanup
  - EntitySystem, 170
- clear
  - Console, 146
  - EntityTracker, 178
  - GameLog, 203
- clear\_buildings
  - BuilderWindow, 124
- clear\_entity\_blueprints
  - WaveSystem, 363
- clear\_freed
  - Grid, 216
- clear\_selected\_entities
  - SelectionBox, 328
- clear\_spawn\_nodes
  - WaveSystem, 363
- clear\_spells
  - SpellCastingWindow, 338
- clear\_stack

- lpp::Script, 321
- clear\_task\_queue
  - TaskHandlerHelper, 96
- clear\_unfreed
  - Grid, 216
- collide
  - GraphicsHelper, 46
- cols\_
  - ResearchWindow, 316
- combat\_
  - TaskSystem, 349
- CombatComponent, 133
- CombatHelper, 20
  - get\_atk\_type, 21
  - get\_cooldown, 21
  - get\_dmg, 22
  - get\_dmg\_range, 22
  - get\_projectile\_blueprint, 22
  - get\_range, 22
  - get\_target, 23
  - in\_range, 23
  - set\_atk\_type, 23
  - set\_cooldown, 23
  - set\_dmg\_range, 24
  - set\_projectile\_blueprint, 24
  - set\_range, 24
  - set\_target, 25
- CombatSystem, 134
  - ~CombatSystem, 136
  - apply\_damage\_to\_entities\_in\_range, 136
  - apply\_effect\_to\_entities\_in\_range, 137
  - apply\_freeze\_to, 137
  - apply\_freeze\_to\_entities\_in\_range, 137
  - apply\_heal\_to\_entities\_in\_range, 137
  - apply\_slow\_to, 138
  - apply\_slow\_to\_entities\_in\_range, 138
  - CombatSystem, 136
  - create\_homing\_projectile, 138
  - enemy\_in\_range, 138
  - entities\_, 143
  - get\_closest\_entity, 139
  - get\_closest\_entity\_thats\_not, 139
  - get\_closest\_gold\_deposit, 140
  - get\_closest\_gold\_vault, 140
  - get\_closest\_structure, 140
  - get\_container, 140, 141
  - get\_max\_run\_away\_attempts, 141
  - grid\_, 143
  - in\_sight, 141
  - in\_sight\_wrt\_BB, 141
  - max\_run\_away\_attempts\_, 143
  - ray\_caster\_, 143
  - ray\_query\_, 143
  - run\_away\_from, 142
  - run\_away\_from\_, 142
  - run\_away\_queue\_, 143
  - set\_max\_run\_away\_attempts, 142
  - update, 142
- command\_to\_attack
  - CommandHelper, 25
- command\_to\_fall\_back
  - CommandHelper, 26
- command\_to\_mine
  - CommandHelper, 26
- command\_to\_reposition
  - CommandHelper, 26
- command\_to\_return\_gold
  - CommandHelper, 26
- CommandComponent, 144
- CommandHelper, 25
  - command\_to\_attack, 25
  - command\_to\_fall\_back, 26
  - command\_to\_mine, 26
  - command\_to\_reposition, 26
  - command\_to\_return\_gold, 26
  - set\_command, 27
  - test\_command, 27
- Component, 145
- Console, 145
  - ~Console, 146
  - clear, 146
  - Console, 146
  - console\_history\_, 148
  - curr\_command\_, 148
  - execute, 146
  - get\_history, 147
  - handle\_text, 147
  - init\_, 147
  - list\_box\_, 149
  - print\_text, 147
  - scroll\_down, 147
  - set\_history, 148
  - set\_visible, 148
  - time\_since\_last\_fps\_update\_, 149
  - update\_fps, 148
- console\_
  - GUI, 232
- console\_history\_
  - Console, 148
- ConstructorComponent, 149
- ConstructorHelper, 27
  - call, 28
  - get\_blueprint, 28
  - set\_blueprint, 28
- constructors\_to\_be\_called\_
  - EntitySystem, 175
- CounterComponent, 150
- CounterHelper, 28
  - decrement, 29
  - get\_curr\_value, 29
  - get\_max\_value, 29
  - increment, 30
  - set\_curr\_value, 30
  - set\_max\_value, 30
- create\_empty\_level
  - Game, 194

- create\_entity
  - EntitySystem, 170
- create\_graph
  - Grid, 216
- create\_graphics
  - GridSystem, 223
- create\_homing\_projectile
  - CombatSystem, 138
- create\_task
  - TaskHelper, 99
- curr\_active\_spell\_
  - SpellCastingWindow, 340
- curr\_command\_
  - Console, 148
- curr\_id\_
  - EntitySystem, 175
- curr\_position\_
  - EntityPlacer, 160
- curr\_spell\_
  - Spellcaster, 335
- curr\_tool\_
  - GUI, 232
- curr\_tracked\_entity\_
  - EntityTracker, 180
- curr\_wave\_number\_
  - WaveSystem, 369
- current\_task\_completed\_
  - TaskSystem, 347
- currently\_binded\_action\_
  - OptionsWindow, 289
- cycles\_
  - level\_generators::LevelGenerator, 258
- DAMAGE\_EFFECT
  - util::effect::DAMAGE\_EFFECT, 151
- DEFAULT\_LEVEL\_GENERATOR
  - level\_generators, 63
- DEFAULT\_PATH\_TYPE
  - util, 113
- deactivate\_current\_spell
  - SpellCastingWindow, 338
- debuff
  - HealthHelper, 56
- dec\_selection
  - BuilderWindow, 124
  - SpellCastingWindow, 338
- decrease
  - ExperienceValueHelper, 37
- decrement
  - CounterHelper, 29
- delete\_component
  - EntitySystem, 171
- delete\_component\_now
  - EntitySystem, 171
- delete\_entities
  - EntitySystem, 172
- delete\_graphics
  - GridSystem, 223
- delete\_input\_
  - InputSystem, 248
- delete\_possible\_event
  - EventHandlerHelper, 33
- delete\_possible\_task
  - TaskHandlerHelper, 96
- destroy
  - DestructorHelper, 31
  - util::EntityDestroyer, 156
- destroy\_entity
  - EntitySystem, 172
- DestructorComponent, 152
- DestructorHelper, 30
  - destroy, 31
  - get\_blueprint, 31
  - set\_blueprint, 31
- dir\_to
  - MovementHelper, 71
- distance\_
  - Grid, 219
- distribute\_to\_adjacent\_free\_nodes
  - Grid, 216
- dummy\_unlock
  - ResearchWindow, 314
- end\_wave
  - WaveSystem, 364
- endless\_mode\_
  - WaveSystem, 369
- enemy\_faction\_
  - util::IS\_ENEMY, 251
- enemy\_in\_range
  - CombatSystem, 138
- ent\_
  - EntityPlacer, 160
- entities\_
  - AISystem, 120
  - CombatSystem, 143
  - EntityPlacer, 160
  - EntitySystem, 176
  - EntityTracker, 181
  - EventSystem, 185
  - GameSerializer, 208
  - GraphicsSystem, 212
  - GridSystem, 224
  - HealthSystem, 243
  - InputSystem, 249
  - level\_generators::LevelGenerator, 258
  - ManaSpellSystem, 275
  - MovementSystem, 282
  - Player, 299
  - ProductionSystem, 308
  - SelectionBox, 330
  - TaskSystem, 349
  - TimeSystem, 353
  - TriggerSystem, 358
  - util::HAS\_GOLD, 238
  - util::IS\_ENEMY, 251
  - util::IS\_FRIENDLY\_OR\_NEUTRAL, 254
  - util::IS\_FRIENDLY, 252

- util::IS\_GOLD\_VAULT, 256
- util::effect::DAMAGE\_EFFECT, 152
- util::effect::FREEZE\_EFFECT, 191
- util::effect::HEAL\_EFFECT, 239
- util::effect::LOWER\_SPEED\_EFFECT, 260
- WaveSystem, 370
- entities\_spawned\_
  - WaveSystem, 370
- entities\_total\_
  - WaveSystem, 370
- entity\_blueprints\_
  - WaveSystem, 370
- entity\_creator\_
  - Game, 199
- entity\_register\_
  - EntitySystem, 176
- entity\_system\_
  - Game, 199
- EntityCreator, 153
  - ~EntityCreator, 154
  - actualize\_list, 154
  - change\_to\_create, 155
  - change\_to\_place, 155
  - EntityCreator, 154
  - init\_, 155
  - list\_box\_, 156
  - place, 155
  - placer\_, 156
  - registered\_entities\_, 156
- EntityPlacer, 157
  - ~EntityPlacer, 159
  - can\_place\_when\_game\_paused, 159
  - can\_place\_when\_game\_paused\_, 160
  - curr\_position\_, 160
  - ent\_, 160
  - entities\_, 160
  - EntityPlacer, 158
  - grid\_, 161
  - half\_height\_, 161
  - is\_visible, 159
  - mgr\_, 161
  - place, 159
  - placing\_node\_, 161
  - placing\_structure\_, 161
  - price\_, 161
  - set\_current\_entity\_table, 159
  - set\_visible, 159
  - structure\_radius\_, 161
  - table\_name\_, 162
  - toggle\_placing\_when\_game\_paused, 160
  - update\_position, 160
  - visible\_, 162
- EntitySystem, 162
  - ~EntitySystem, 169
  - add\_component, 169
  - clean\_up\_component, 170
  - cleanup, 170
  - constructors\_to\_be\_called\_, 175
  - create\_entity, 170
  - curr\_id\_, 175
  - delete\_component, 171
  - delete\_component\_now, 171
  - delete\_entities, 172
  - destroy\_entity, 172
  - entities\_, 176
  - entity\_register\_, 176
  - EntitySystem, 169
  - exists, 172
  - FACTION\_NAME, 176
  - get\_component, 172
  - get\_component\_container, 173
  - get\_component\_list, 173
  - get\_new\_id, 173
  - get\_registered\_entities, 173
  - get\_scene\_manager, 173
  - has\_component, 173, 174
  - init\_function\_arrays, 174
  - load\_component, 174
  - loaders\_, 176
  - NO\_BLUEPRINT, 176
  - physics\_, 176
  - register\_entity, 174
  - scene\_, 176
  - set\_component, 175
  - to\_be\_destroyed\_, 177
  - update, 175
- EntityTracker, 177
  - ~EntityTracker, 178
  - clear, 178
  - curr\_tracked\_entity\_, 180
  - entities\_, 181
  - EntityTracker, 178
  - get\_tracked\_entity, 178
  - init\_, 178
  - init\_upgrade\_but, 179
  - set\_tracked\_entity, 180
  - show\_upgrade\_but, 180
  - update\_tracking, 180
- escape\_pressed
  - GUI, 228
- EventComponent, 181
- EventHandlerComponent, 182
- EventHandlerHelper, 32
  - add\_possible\_event, 32
  - can\_handle, 32
  - delete\_possible\_event, 33
  - get\_handler, 33
  - set\_handler, 33
- EventHelper, 34
  - get\_event\_handler, 34
  - get\_event\_type, 34
  - get\_radius, 35
  - get\_target, 35
  - is\_active, 35
  - set\_active, 35
  - set\_event\_handler, 36



- set\_event\_type, 36
- set\_radius, 36
- set\_target, 37
- EventSystem, 183
  - ~EventSystem, 184
  - entities\_, 185
  - EventSystem, 183
  - get\_update\_period, 184
  - get\_update\_time\_multiplier, 184
  - handle\_event\_, 184
  - set\_update\_period, 184
  - set\_update\_time\_multiplier, 185
  - update, 185
  - update\_period\_, 185
  - update\_time\_multiplier\_, 185
- Exception
  - lpp::Exception, 186
- execute
  - Console, 146
  - lpp::Script, 321
- execute\_selection
  - SelectionBox, 328
- execute\_single\_selection
  - SelectionBox, 328
- exists
  - EntitySystem, 172
- ExperienceValueComponent, 187
- ExperienceValueHelper, 37
  - decrease, 37
  - get, 38
  - increase, 38
  - set, 38
- ExplosionComponent, 188
- ExplosionHelper, 38
  - get\_curr\_radius, 39
  - get\_delta, 39
  - get\_max\_radius, 39
  - increase\_curr\_radius, 40
  - set\_delta, 40
  - set\_max\_radius, 40
- extend\_to
  - SelectionBox, 329
- FACTION\_NAME
  - EntitySystem, 176
- FREEZE\_EFFECT
  - util::effect::FREEZE\_EFFECT, 190
- faction\_
  - util::IS\_FRIENDLY\_OR\_NEUTRAL, 254
  - util::IS\_FRIENDLY, 253
- FactionComponent, 189
- FactionHelper, 41
  - get\_faction, 41
  - get\_faction\_name, 41
  - set\_faction, 41
- file\_
  - GameSerializer, 208
- first\_person\_
  - InputSystem, 249
- first\_person\_id\_
  - InputSystem, 249
- force\_update
  - AISystem, 120
- frameRenderingQueued
  - Game, 195
- free\_mode\_
  - Camera, 132
- free\_nodes\_
  - Grid, 219
- free\_research
  - ResearchWindow, 314
- freed\_
  - Grid, 220
- fullscreen\_
  - OptionsWindow, 289
- GUIWindow, 234
  - ~GUIWindow, 235
  - GUIWindow, 235
  - init, 236
  - init\_, 236
  - is\_visible, 236
  - set\_visible, 236
  - window\_, 236
- GUI, 225
  - ~GUI, 227
  - builder\_, 232
  - console\_, 232
  - curr\_tool\_, 232
  - escape\_pressed, 228
  - GUI, 227
  - game\_, 232
  - get\_builder, 228
  - get\_console, 228
  - get\_curr\_tool, 228
  - get\_log, 228
  - get\_message, 228
  - get\_options, 229
  - get\_research, 229
  - get\_spell\_casting, 229
  - get\_top\_bar, 229
  - get\_tracker, 229
  - get\_window, 229
  - init, 230
  - instance, 230
  - is\_visible, 230
  - list\_directory, 230
  - log\_, 232
  - menu\_, 233
  - message\_, 233
  - options\_, 233
  - research\_, 233
  - set\_curr\_tool, 231
  - set\_curr\_tool\_visible, 231
  - set\_visible, 231
  - show\_load\_save\_dialog, 232
  - spell\_casting\_, 233
  - top\_bar\_, 233

- tracker\_, 233
  - window\_, 233
- Game, 192
  - ~Game, 194
  - create\_empty\_level, 194
  - entity\_creator\_, 199
  - entity\_system\_, 199
  - frameRenderingQueued, 195
  - Game, 194
  - game\_serializer\_, 199
  - get\_mouse\_click\_position, 195
  - get\_throne\_id, 195
  - ground\_, 199
  - ground\_entity\_, 199
  - keyPressed, 195
  - keyReleased, 195
  - level\_generator\_, 200
  - main\_cam\_, 200
  - mouse\_position\_, 200
  - mouseMoved, 195
  - mousePressed, 195
  - mouseReleased, 196
  - new\_game, 196
  - ogre\_init, 196
  - ois\_to\_cegui, 196
  - placer\_, 200
  - renderer\_, 200
  - reset\_camera, 196
  - reset\_unlocks, 197
  - root\_, 200
  - run, 197
  - selection\_box\_, 200
  - set\_state, 197
  - set\_throne\_id, 197
  - spell\_caster\_, 201
  - state\_, 201
  - systems\_, 201
  - throne\_id\_, 201
  - toggle\_camera\_free\_mode, 197
  - update, 197
  - windowClosed, 199
  - windowResized, 199
- game\_
  - GUI, 232
- game\_serializer\_
  - Game, 199
- GameLog, 201
  - ~GameLog, 202
  - clear, 203
  - GameLog, 202
  - get\_history, 203
  - init\_, 203
  - log\_, 204
  - log\_history\_, 204
  - print, 203
  - set\_history, 203
- GameSerializer, 204
  - ~GameSerializer, 207
- entities\_, 208
- file\_, 208
- GameSerializer, 207
- Grid, 219
- load\_game, 207
- save\_component, 207
- save\_entities\_, 208
- save\_game, 208
- save\_tasks, 208
- save\_unlocks, 208
- save\_wave\_system, 208
- script\_, 209
- serializers\_, 209
- task\_pairs\_, 209
- generate
  - level\_generators::LevelGenerator, 257
  - level\_generators::RandomLevelGenerator, 310
- get
  - ExperienceValueHelper, 38
  - lpp::Script, 321
- get\_
  - lpp::Script, 321
- get\_2d\_position
  - PhysicsHelper, 81
- get\_angle
  - PhysicsHelper, 82
- get\_atk\_type
  - CombatHelper, 21
- get\_blueprint
  - AIHelper, 19
  - ConstructorHelper, 28
  - DestructorHelper, 31
  - OnHitHelper, 78
  - SpellHelper, 91
  - TaskHandlerHelper, 96
  - TriggerHelper, 105
  - UpgradeHelper, 109
- get\_board\_coords
  - GridNodeHelper, 52
- get\_bounds
  - GraphicsHelper, 46
- get\_builder
  - GUI, 228
- get\_building\_
  - BuilderWindow, 124
- get\_buildings
  - BuilderWindow, 125
- get\_capacity
  - ManaCrystalHelper, 67
- get\_center\_position
  - Grid, 217
- get\_check\_period
  - TriggerSystem, 357
- get\_closest\_entity
  - CombatSystem, 139
- get\_closest\_entity\_thats\_not
  - CombatSystem, 139
- get\_closest\_gold\_deposit

- CombatSystem, [140](#)
- get\_closest\_gold\_vault
  - CombatSystem, [140](#)
- get\_closest\_portal
  - util::heuristic::PORTAL\_HEURISTIC, [301](#)
- get\_closest\_structure
  - CombatSystem, [140](#)
- get\_component
  - EntitySystem, [172](#)
- get\_component\_container
  - EntitySystem, [173](#)
- get\_component\_list
  - EntitySystem, [173](#)
- get\_console
  - GUI, [228](#)
- get\_container
  - CombatSystem, [140](#), [141](#)
- get\_cooldown
  - CombatHelper, [21](#)
  - NotificationHelper, [76](#)
  - OnHitHelper, [78](#)
  - SpellHelper, [91](#)
  - TriggerHelper, [106](#)
- get\_cost
  - PathfindingHelper, [79](#)
- get\_countdown\_value
  - WaveSystem, [364](#)
- get\_curr\_gold
  - GoldHelper, [43](#)
- get\_curr\_radius
  - ExplosionHelper, [39](#)
- get\_curr\_time
  - LimitedLifeSpanHelper, [66](#)
  - NotificationHelper, [76](#)
  - SpellHelper, [91](#)
  - TimeHelper, [102](#)
- get\_curr\_tool
  - GUI, [228](#)
- get\_curr\_value
  - CounterHelper, [29](#)
- get\_curr\_wave\_number
  - WaveSystem, [364](#)
- get\_defense
  - HealthHelper, [56](#)
- get\_delta
  - ExplosionHelper, [39](#)
- get\_dir
  - MovementHelper, [72](#)
- get\_dir\_back
  - MovementHelper, [72](#)
- get\_dir\_left
  - MovementHelper, [72](#)
- get\_dir\_right
  - MovementHelper, [72](#)
- get\_direction
  - Camera, [128](#)
- get\_distance
  - Grid, [217](#)
- PhysicsHelper, [82](#)
- get\_dmg
  - CombatHelper, [22](#)
  - HomingHelper, [60](#)
- get\_dmg\_range
  - CombatHelper, [22](#)
- get\_endless\_mode
  - WaveSystem, [364](#)
- get\_entities\_spawned
  - WaveSystem, [364](#)
- get\_entity\_blueprints
  - WaveSystem, [364](#)
- get\_entity\_total
  - WaveSystem, [364](#)
- get\_enum\_direction
  - util, [114](#)
- get\_event\_handler
  - EventHelper, [34](#)
- get\_event\_type
  - EventHelper, [34](#)
- get\_exp\_needed
  - UpgradeHelper, [109](#)
- get\_experience
  - UpgradeHelper, [110](#)
- get\_faction
  - FactionHelper, [41](#)
- get\_faction\_name
  - FactionHelper, [41](#)
- get\_field\_to\_stack
  - lpp::Script, [321](#)
- get\_free\_mode
  - Camera, [128](#)
- get\_freed
  - Grid, [217](#)
- get\_gold
  - Player, [296](#)
- get\_half\_height
  - PhysicsHelper, [82](#)
- get\_handler
  - EventHandlerHelper, [33](#)
- get\_health
  - HealthHelper, [57](#)
- get\_height
  - Camera, [128](#)
- get\_history
  - Console, [147](#)
  - GameLog, [203](#)
- get\_info
  - RayCaster, [312](#)
- get\_initial\_buildings
  - Player, [296](#)
- get\_initial\_spells
  - Player, [296](#)
- get\_input\_handler
  - InputHelper, [62](#)
- get\_key\_bind\_name\_
  - OptionsWindow, [288](#)
- get\_last\_spell

- Spellcaster, [334](#)
- get\_last\_spell\_id
  - Spellcaster, [334](#)
- get\_last\_spell\_type
  - Spellcaster, [334](#)
- get\_level
  - UpgradeHelper, [110](#)
- get\_level\_cap
  - UpgradeHelper, [110](#)
- get\_linked\_entity
  - TriggerHelper, [106](#)
- get\_log
  - GUI, [228](#)
- get\_mana
  - ManaHelper, [69](#)
  - Player, [296](#)
- get\_mana\_regen
  - Player, [296](#)
- get\_manhattan\_distance
  - GridNodeHelper, [52](#)
- get\_manual\_scaling
  - GraphicsHelper, [47](#)
- get\_material
  - GraphicsHelper, [47](#)
- get\_max\_gold
  - GoldHelper, [43](#)
- get\_max\_mana
  - ManaHelper, [69](#)
  - Player, [297](#)
- get\_max\_radius
  - ExplosionHelper, [39](#)
- get\_max\_run\_away\_attempts
  - CombatSystem, [141](#)
- get\_max\_time
  - LimitedLifeSpanHelper, [66](#)
- get\_max\_value
  - CounterHelper, [29](#)
- get\_mesh
  - GraphicsHelper, [47](#)
- get\_message
  - GUI, [228](#)
- get\_mouse\_click\_position
  - Game, [195](#)
- get\_name
  - NameHelper, [74](#)
- get\_neighbours
  - GridNodeHelper, [52](#)
- get\_new\_id
  - EntitySystem, [173](#)
- get\_node
  - Grid, [217](#)
- get\_node\_from\_position
  - Grid, [217](#)
- get\_node\_in\_dir
  - GridNodeHelper, [52](#)
- get\_options
  - GUI, [229](#)
- get\_original\_speed
  - MovementHelper, [73](#)
- get\_path
  - PathfindingHelper, [80](#)
- get\_pathfinding\_blueprint
  - PathfindingHelper, [80](#)
- get\_position
  - Camera, [129](#)
  - PhysicsHelper, [82](#)
- get\_price
  - PriceHelper, [85](#)
- get\_price\_
  - ResearchWindow, [314](#)
- get\_producer
  - ProductHelper, [86](#)
- get\_production\_blueprint
  - ProductionHelper, [87](#)
- get\_production\_cooldown
  - ProductionHelper, [87](#)
- get\_production\_count
  - ProductionHelper, [88](#)
- get\_production\_limit
  - ProductionHelper, [88](#)
- get\_production\_progress
  - ProductionHelper, [88](#)
- get\_projectile\_blueprint
  - CombatHelper, [22](#)
- get\_query\_flags
  - GraphicsHelper, [47](#)
- get\_radius
  - EventHelper, [35](#)
  - StructureHelper, [94](#)
  - TriggerHelper, [106](#)
- get\_random
  - util, [114](#)
- get\_random\_free\_node
  - Grid, [218](#)
- get\_range
  - CombatHelper, [22](#)
- get\_regen
  - HealthHelper, [57](#)
  - ManaCrystalHelper, [67](#)
  - ManaHelper, [69](#)
- get\_regen\_period
  - HealthSystem, [242](#)
  - ManaSpellSystem, [274](#)
- get\_registered\_entities
  - EntitySystem, [173](#)
- get\_research
  - GUI, [229](#)
- get\_resident
  - GridNodeHelper, [53](#)
- get\_scale
  - GraphicsHelper, [48](#)
- get\_scene\_manager
  - EntitySystem, [173](#)
- get\_selected\_entities
  - SelectionBox, [329](#)
- get\_source

- HomingHelper, 60
- get\_spawn\_cooldown
  - WaveSystem, 364
- get\_spawn\_timer
  - WaveSystem, 365
- get\_spawning\_nodes
  - WaveSystem, 365
- get\_speed\_modifier
  - MovementHelper, 73
- get\_spell
  - Spellcaster, 334
- get\_spell\_
  - SpellCastingWindow, 339
- get\_spell\_casting
  - GUI, 229
- get\_spell\_type
  - Spellcaster, 334
- get\_spells
  - SpellCastingWindow, 339
- get\_stack\_contents
  - lpp::Script, 323
- get\_state
  - AIHelper, 20
  - lpp::Script, 323
  - WaveSystem, 365
- get\_target
  - CombatHelper, 23
  - EventHelper, 35
  - HomingHelper, 61
  - TimeHelper, 103
- get\_task\_name
  - TaskSystem, 348
- get\_task\_queue
  - TaskHandlerHelper, 96
- get\_task\_source
  - TaskHelper, 99
- get\_task\_target
  - TaskHelper, 99
- get\_task\_type
  - TaskHelper, 99
- get\_throne\_id
  - Game, 195
- get\_time\_limit
  - TimeHelper, 103
- get\_time\_multiplier
  - ProductionSystem, 306
  - TimeSystem, 352
- get\_top\_bar
  - GUI, 229
- get\_tracked\_entity
  - EntityTracker, 178
- get\_tracker
  - GUI, 229
- get\_type
  - TimeHelper, 103
- get\_unfreed
  - Grid, 218
- get\_unlocked
  - ResearchWindow, 315
- get\_update\_period
  - AISystem, 120
  - EventSystem, 184
  - GraphicsSystem, 212
- get\_update\_time\_multiplier
  - EventSystem, 184
- get\_vector
  - lpp::Script, 323
- get\_wave\_count
  - WaveSystem, 365
- get\_wave\_entities
  - WaveSystem, 365
- get\_wave\_table
  - WaveSystem, 365
- get\_window
  - GUI, 229
- gold\_
  - Player, 299
- gold\_full
  - GoldHelper, 43
- GoldComponent, 209
- GoldHelper, 42
  - add\_gold, 42
  - get\_curr\_gold, 43
  - get\_max\_gold, 43
  - gold\_full, 43
  - set\_curr\_gold, 43
  - set\_max\_gold, 44
  - sub\_gold, 44
  - transfer\_all\_gold, 44
- graphics\_loaded\_
  - GridSystem, 224
- GraphicsComponent, 210
- GraphicsHelper, 45
  - apply\_scale, 46
  - collide, 46
  - get\_bounds, 46
  - get\_manual\_scaling, 47
  - get\_material, 47
  - get\_mesh, 47
  - get\_query\_flags, 47
  - get\_scale, 48
  - init\_graphics\_component, 48
  - is\_visible, 48
  - look\_at, 48
  - PLANE, 46
  - rotate, 49
  - set\_manual\_scaling, 49
  - set\_material, 49
  - set\_mesh, 50
  - set\_query\_flags, 50
  - set\_scale, 50
  - set\_visible, 50
- GraphicsSystem, 211
  - ~GraphicsSystem, 212
  - entities\_, 212
  - get\_update\_period, 212

- GraphicsSystem, 211
- set\_update\_period, 212
- update, 212
- update\_timer\_, 212
- Grid, 213
  - ~Grid, 215
  - add\_freed, 215
  - add\_node, 215
  - add\_unfreed, 216
  - clear\_freed, 216
  - clear\_unfreed, 216
  - create\_graph, 216
  - distance\_, 219
  - distribute\_to\_adjacent\_free\_nodes, 216
  - free\_nodes\_, 219
  - freed\_, 220
  - GameSerializer, 219
  - get\_center\_position, 217
  - get\_distance, 217
  - get\_freed, 217
  - get\_node, 217
  - get\_node\_from\_position, 217
  - get\_random\_free\_node, 218
  - get\_unfreed, 218
  - Grid, 215
  - in\_board, 218
  - instance, 218
  - link\_, 218
  - nodes\_, 220
  - place\_at\_random\_free\_node, 219
  - remove\_node, 219
  - start\_, 220
  - starting\_index\_, 220
  - width\_, 220
- grid\_
  - CombatSystem, 143
  - EntityPlacer, 161
  - ProductionSystem, 308
  - TaskSystem, 349
- GridNodeComponent, 221
- GridNodeHelper, 51
  - area\_free, 51
  - get\_board\_coords, 52
  - get\_manhattan\_distance, 52
  - get\_neighbours, 52
  - get\_node\_in\_dir, 52
  - get\_resident, 53
  - is\_free, 53
  - set\_free, 53
  - set\_free\_selected, 53
  - set\_portal\_neighbour, 54
  - set\_resident, 54
- GridSystem, 222
  - ~GridSystem, 223
  - create\_graphics, 223
  - delete\_graphics, 223
  - entities\_, 224
  - graphics\_loaded\_, 224
  - GridSystem, 223
  - is\_visible, 223
  - place\_structure, 223
  - scene\_mgr\_, 225
  - set\_visible, 224
  - update, 224
  - update\_neighbours\_, 224
- ground\_
  - Game, 199
- ground\_entity\_
  - Game, 199
- HAS\_GOLD
  - util::HAS\_GOLD, 237
- HEAL\_EFFECT
  - util::effect::HEAL\_EFFECT, 239
- half\_height\_
  - EntityPlacer, 161
- handle\_event\_
  - EventSystem, 184
  - TimeSystem, 352
- handle\_task\_
  - TaskSystem, 348
- handle\_text
  - Console, 147
- has\_component
  - EntitySystem, 173, 174
- has\_lua\_state
  - lpp::Exception, 186
- heal
  - HealthHelper, 57
- HealthComponent, 240
- HealthHelper, 54
  - add\_defense, 55
  - add\_health, 56
  - buff, 56
  - debuff, 56
  - get\_defense, 56
  - get\_health, 57
  - get\_regen, 57
  - heal, 57
  - is\_alive, 57
  - set\_alive, 58
  - set\_defense, 58
  - set\_health, 58
  - set\_regen, 58
  - sub\_defense, 59
  - sub\_health, 59
  - ubercharge, 59
- HealthSystem, 240
  - ~HealthSystem, 241
  - entities\_, 243
  - get\_regen\_period, 242
  - HealthSystem, 241
  - regen\_, 243
  - regen\_period\_, 243
  - regen\_timer\_, 243
  - set\_regen\_period, 242
  - update, 242

- update\_regen, 242
- height\_
  - Camera, 132
- HomingComponent, 244
- HomingHelper, 60
  - get\_dmg, 60
  - get\_source, 60
  - get\_target, 61
  - set\_dmg, 61
  - set\_source, 61
  - set\_target, 61
- IS\_ENEMY
  - util::IS\_ENEMY, 250
- IS\_FRIENDLY\_OR\_NEUTRAL
  - util::IS\_FRIENDLY\_OR\_NEUTRAL, 253
- IS\_FRIENDLY
  - util::IS\_FRIENDLY, 252
- IS\_GOLD\_VAULT
  - util::IS\_GOLD\_VAULT, 255
- in\_board
  - Grid, 218
- in\_range
  - CombatHelper, 23
- in\_sight
  - CombatSystem, 141
- in\_sight\_wrt\_BB
  - CombatSystem, 141
- inc\_selection
  - BuilderWindow, 125
  - SpellCastingWindow, 339
- increase
  - ExperienceValueHelper, 38
- increase\_curr\_radius
  - ExplosionHelper, 40
- increment
  - CounterHelper, 30
- init
  - Camera, 129
  - GUIWindow, 236
  - GUI, 230
  - LightHelper, 64
  - LuaInterface, 271
  - Player, 297
- init\_
  - BuilderWindow, 125
  - Console, 147
  - EntityCreator, 155
  - EntityTracker, 178
  - GUIWindow, 236
  - GameLog, 203
  - MessageToPlayerWindow, 277
  - OptionsWindow, 288
  - ResearchWindow, 315
  - SpellCastingWindow, 339
  - TopBar, 354
- init\_function\_arrays
  - EntitySystem, 174
- init\_graphics\_component
  - GraphicsHelper, 48
- init\_upgrade\_butt
  - EntityTracker, 179
- initial\_building\_unlocks\_
  - Player, 299
- initial\_spell\_unlocks\_
  - Player, 300
- InputComponent, 245
- InputHelper, 62
  - get\_input\_handler, 62
  - set\_input\_handler, 62
- InputSystem, 246
  - ~InputSystem, 247
  - ai\_backup\_, 248
  - cam\_, 248
  - cam\_position\_, 248
  - delete\_input\_, 248
  - entities\_, 249
  - first\_person\_, 249
  - first\_person\_id\_, 249
  - InputSystem, 247
  - is\_first\_person, 247
  - KEY\_UP, 249
  - keyboard\_, 249
  - rebind, 247
  - set\_first\_person, 248
  - task\_backup\_, 249
  - update, 248
- instance
  - GUI, 230
  - Grid, 218
  - lpp::Script, 323
  - Player, 297
- is\_active
  - EventHelper, 35
- is\_alive
  - HealthHelper, 57
- is\_casting
  - Spellcaster, 334
- is\_complete
  - TaskHelper, 100
- is\_first\_person
  - InputSystem, 247
- is\_free
  - GridNodeHelper, 53
- is\_nil
  - lpp::Script, 323
- is\_selecting
  - SelectionBox, 329
- is\_solid
  - PhysicsHelper, 83
- is\_unlocked\_
  - ResearchWindow, 315
- is\_visible
  - EntityPlacer, 159
  - GUIWindow, 236
  - GUI, 230
  - GraphicsHelper, 48

- GridSystem, 223
- LightHelper, 64
- is\_walk\_through
  - StructureHelper, 94
- KEY\_UP
  - InputSystem, 249
- key\_binds\_
  - OptionsWindow, 289
- key\_names\_
  - OptionsWindow, 290
- key\_pressed
  - Camera, 129
  - OptionsWindow, 288
- key\_released
  - Camera, 129
- keyPressed
  - Game, 195
- keyReleased
  - Game, 195
- keyboard\_
  - InputSystem, 249
- L
  - lpp::Script, 325
- L\_
  - lpp::Exception, 187
- LOWER\_SPEED\_EFFECT
  - util::effect::LOWER\_SPEED\_EFFECT, 260
- label\_text\_
  - WaveSystem, 370
- last\_delta\_
  - MovementSystem, 282
- last\_spell\_
  - Spellcaster, 335
- last\_spell\_id\_
  - Spellcaster, 335
- level\_generator\_
  - Game, 200
- level\_generators, 63
  - DEFAULT\_LEVEL\_GENERATOR, 63
- level\_generators::LevelGenerator, 256
  - ~LevelGenerator, 257
  - cycles\_, 258
  - entities\_, 258
  - generate, 257
  - LevelGenerator, 257
- level\_generators::RandomLevelGenerator, 309
  - generate, 310
  - RandomLevelGenerator, 309
- LevelGenerator
  - level\_generators::LevelGenerator, 257
- LightComponent, 258
- LightHelper, 63
  - init, 64
  - is\_visible, 64
  - set\_visible, 64
  - toggle\_visible, 65
- LimitedLifeSpanComponent, 259
- LimitedLifeSpanHelper, 65
  - advance\_curr\_time, 65
  - get\_curr\_time, 66
  - get\_max\_time, 66
  - set\_max\_time, 66
- link\_
  - Grid, 218
- list\_box\_
  - Console, 149
  - EntityCreator, 156
- list\_directory
  - GUI, 230
- load
  - lpp::Script, 323
- load\_component
  - EntitySystem, 174
- load\_game
  - GameSerializer, 207
- loaded\_scripts\_
  - lpp::Script, 325
- loaders\_
  - EntitySystem, 176
- log\_
  - GUI, 232
  - GameLog, 204
- log\_history\_
  - GameLog, 204
- look\_at
  - Camera, 129
  - GraphicsHelper, 48
- lpp::Exception, 186
  - Exception, 186
  - has\_lua\_state, 186
  - L\_, 187
  - msg\_, 187
  - what, 187
  - what\_lua, 187
- lpp::Script, 318
  - ~Script, 320
  - call, 320, 321
  - clear\_stack, 321
  - execute, 321
  - get, 321
  - get\_, 321
  - get\_field\_to\_stack, 321
  - get\_stack\_contents, 323
  - get\_state, 323
  - get\_vector, 323
  - instance, 323
  - is\_nil, 323
  - L, 325
  - load, 323
  - loaded\_scripts\_, 325
  - push\_arg, 324
  - push\_args, 324
  - register\_function, 325
  - reload\_all\_scripts, 325
  - Script, 320



- set, 325
- lua\_get\_avg\_fps
  - LuaInterface, 271
- lua\_this
  - LuaInterface, 271
- LuaInterface, 261
  - init, 271
  - lua\_get\_avg\_fps, 271
  - lua\_this, 271
  - LuaInterface, 271
- MENU\_TAB
  - action, 18
- main\_cam\_
  - Game, 200
- mana\_
  - Player, 300
- mana\_regen\_
  - Player, 300
- ManaComponent, 272
- ManaCrystalComponent, 272
- ManaCrystalHelper, 66
  - get\_capacity, 67
  - get\_regen, 67
  - set\_capacity, 67
  - set\_regen, 68
- ManaHelper, 68
  - add\_mana, 69
  - get\_mana, 69
  - get\_max\_mana, 69
  - get\_regen, 69
  - set\_mana, 70
  - set\_max\_mana, 70
  - set\_regen, 70
  - sub\_mana, 70
- ManaSpellSystem, 273
  - ~ManaSpellSystem, 274
  - entities\_, 275
  - get\_regen\_period, 274
  - ManaSpellSystem, 274
  - regen\_timer\_, 275
  - set\_regen\_period, 274
  - update, 275
- max\_curr\_time
  - TimeHelper, 103
- max\_mana\_
  - Player, 300
- max\_run\_away\_attempts\_
  - CombatSystem, 143
- menu\_
  - GUI, 233
- message\_
  - GUI, 233
- MessageToPlayerWindow, 276
  - ~MessageToPlayerWindow, 277
  - init\_, 277
  - MessageToPlayerWindow, 277
  - ok\_func\_, 278
  - reset\_but labels, 277
- script\_, 278
  - set\_but\_label, 277
  - show, 278
- mgr\_
  - EntityPlacer, 161
- min\_
  - util::effect::DAMAGE\_EFFECT, 152
- MineComponent, 279
- mouse\_position\_
  - Game, 200
- mouseMoved
  - Game, 195
- mousePressed
  - Game, 195
- mouseReleased
  - Game, 196
- move
  - Camera, 130
  - MovementSystem, 282
- move\_to
  - PhysicsHelper, 83
- movement\_direction\_
  - Camera, 132
- MovementComponent, 279
- MovementHelper, 71
  - dir\_to, 71
  - get\_dir, 72
  - get\_dir\_back, 72
  - get\_dir\_left, 72
  - get\_dir\_right, 72
  - get\_original\_speed, 73
  - get\_speed\_modifier, 73
  - reset\_speed, 73
  - set\_original\_speed, 73
  - set\_speed\_modifier, 74
- MovementSystem, 280
  - ~MovementSystem, 281
  - can\_move\_to, 281
  - checked\_move, 281
  - entities\_, 282
  - last\_delta\_, 282
  - move, 282
  - MovementSystem, 281
  - update, 282
- msg\_
  - lpp::Exception, 187
- NEXT
  - action, 18
- NO\_BLUEPRINT
  - EntitySystem, 176
- NameComponent, 283
- NameHelper, 74
  - get\_name, 74
  - set\_name, 74
- new\_game
  - Game, 196
- next\_task\_
  - TaskSystem, 348

- next\_wave
  - WaveSystem, 365
- next\_wave\_countdown\_
  - WaveSystem, 370
- nodes\_
  - Grid, 220
- NotificationComponent, 284
- NotificationHelper, 75
  - advance\_curr\_time, 75
  - get\_cooldown, 76
  - get\_curr\_time, 76
  - notify, 76
  - reset, 76
  - set\_cooldown, 77
- notify
  - NotificationHelper, 76
- nulify\_all\_stats
  - Player, 297
- ogre\_init
  - Game, 196
- ois\_to\_cegui
  - Game, 196
- ok\_func\_
  - MessageToPlayerWindow, 278
- OnHitComponent, 285
- OnHitHelper, 77
  - call, 77
  - get\_blueprint, 78
  - get\_cooldown, 78
  - set\_blueprint, 78
  - set\_cooldown, 78
- operator()
  - util::HAS\_GOLD, 238
  - util::IS\_ENEMY, 251
  - util::IS\_FRIENDLY\_OR\_NEUTRAL, 254
  - util::IS\_FRIENDLY, 252
  - util::IS\_GOLD\_VAULT, 256
  - util::effect::DAMAGE\_EFFECT, 152
  - util::effect::FREEZE\_EFFECT, 191
  - util::effect::HEAL\_EFFECT, 239
  - util::effect::LOWER\_SPEED\_EFFECT, 260
- options\_
  - GUI, 233
- OptionsWindow, 286
  - ~OptionsWindow, 287
  - actions\_, 289
  - add\_start\_parameters, 288
  - apply\_, 288
  - currently\_bound\_action\_, 289
  - fullscreen\_, 289
  - get\_key\_bind\_name\_, 288
  - init\_, 288
  - key\_binds\_, 289
  - key\_names\_, 290
  - key\_pressed, 288
  - OptionsWindow, 287
  - render\_window\_, 290
  - renderer\_, 290
  - resolutions\_, 290
  - set\_key\_bind, 289
  - update\_labels\_, 289
  - view\_, 290
  - width\_, 290
- PLANE
  - GraphicsHelper, 46
- PREV
  - action, 18
- pathfind
  - util, 114
- PathfindingComponent, 291
- PathfindingHelper, 79
  - can\_break, 79
  - get\_cost, 79
  - get\_path, 80
  - get\_pathfinding\_blueprint, 80
  - set\_pathfinding\_blueprint, 80
- pause
  - WaveSystem, 365
- physics\_
  - EntitySystem, 176
- PhysicsComponent, 291
- PhysicsHelper, 81
  - get\_2d\_position, 81
  - get\_angle, 82
  - get\_distance, 82
  - get\_half\_height, 82
  - get\_position, 82
  - is\_solid, 83
  - move\_to, 83
  - set\_2d\_position, 83
  - set\_half\_height, 84
  - set\_position, 84
  - set\_solid, 84
- pitch
  - Camera, 130
- place
  - EntityCreator, 155
  - EntityPlacer, 159
- place\_at\_random\_free\_node
  - Grid, 219
- place\_structure
  - GridSystem, 223
- placer\_
  - BuilderWindow, 126
  - EntityCreator, 156
  - Game, 200
  - Spellcaster, 336
- placing\_node\_
  - EntityPlacer, 161
- placing\_structure\_
  - EntityPlacer, 161
- Player, 292
  - ~Player, 294
  - add\_curr\_unit, 295
  - add\_gold, 295
  - add\_mana, 295

- add\_mana\_regen, 295
- add\_max\_mana, 296
- add\_max\_unit, 296
- entities\_, 299
- get\_gold, 296
- get\_initial\_buildings, 296
- get\_initial\_spells, 296
- get\_mana, 296
- get\_mana\_regen, 296
- get\_max\_mana, 297
- gold\_, 299
- init, 297
- initial\_building\_unlocks\_, 299
- initial\_spell\_unlocks\_, 300
- instance, 297
- mana\_, 300
- mana\_regen\_, 300
- max\_mana\_, 300
- nulify\_all\_stats, 297
- Player, 294
- reset, 297
- set\_initial\_unlocks, 297
- sub\_curr\_unit, 298
- sub\_gold, 298
- sub\_mana, 298
- sub\_mana\_regen, 298
- sub\_max\_mana, 299
- sub\_max\_unit, 299
- uint\_max\_, 300
- units\_curr\_, 300
- units\_max\_, 300
- PortalComponent, 302
- price\_
  - EntityPlacer, 161
- PriceComponent, 302
- PriceHelper, 85
  - get\_price, 85
  - set\_price, 85
- prices\_
  - ResearchWindow, 316
- print
  - GameLog, 203
- print\_text
  - Console, 147
- ProductComponent, 303
- ProductHelper, 85
  - get\_producer, 86
  - set\_producer, 86
- ProductionComponent, 304
- ProductionHelper, 86
  - get\_production\_blueprint, 87
  - get\_production\_cooldown, 87
  - get\_production\_count, 88
  - get\_production\_limit, 88
  - get\_production\_progress, 88
  - set\_production\_blueprint, 88
  - set\_production\_cooldown, 89
  - set\_production\_count, 89
  - set\_production\_limit, 89
  - set\_production\_progress, 89
- ProductionSystem, 305
  - ~ProductionSystem, 306
  - entities\_, 308
  - get\_time\_multiplier, 306
  - grid\_, 308
  - ProductionSystem, 305
  - set\_time\_multiplier, 306
  - spawn\_entity, 306
  - time\_multiplier\_, 308
  - update, 306
- push\_arg
  - lpp::Script, 324
- push\_args
  - lpp::Script, 324
- QUICK\_LOAD
  - action, 19
- QUICK\_SAVE
  - action, 19
- query\_
  - RayCaster, 312
- RESET\_CAMERA
  - action, 19
- RandomLevelGenerator
  - level\_generators::RandomLevelGenerator, 309
- ray\_caster\_
  - CombatSystem, 143
- ray\_query\_
  - CombatSystem, 143
  - SelectionBox, 330
- RayCaster, 310
  - cast, 311
  - get\_info, 312
  - query\_, 312
  - RayCaster, 311
- rebind
  - InputSystem, 247
- regen\_
  - HealthSystem, 243
- regen\_period\_
  - HealthSystem, 243
- regen\_timer\_
  - HealthSystem, 243
  - ManaSpellSystem, 275
- register\_building
  - BuilderWindow, 125
- register\_entity
  - EntitySystem, 174
- register\_function
  - lpp::Script, 325
- register\_spell
  - SpellCastingWindow, 339
- registered\_entities\_
  - EntityCreator, 156
- reload\_all\_scripts
  - lpp::Script, 325

- remove\_node
  - Grid, [219](#)
- render\_window\_
  - OptionsWindow, [290](#)
- renderer\_
  - Game, [200](#)
  - OptionsWindow, [290](#)
- research\_
  - GUI, [233](#)
- research\_all
  - ResearchWindow, [315](#)
- ResearchWindow, [312](#)
  - ~ResearchWindow, [314](#)
  - cols\_, [316](#)
  - dummy\_unlock, [314](#)
  - free\_research, [314](#)
  - get\_price\_, [314](#)
  - get\_unlocked, [315](#)
  - init\_, [315](#)
  - is\_unlocked\_, [315](#)
  - prices\_, [316](#)
  - research\_all, [315](#)
  - ResearchWindow, [314](#)
  - reset\_research, [316](#)
  - rows\_, [316](#)
  - script\_, [317](#)
  - show, [316](#)
  - unlock, [316](#)
  - unlocked\_, [317](#)
- reset
  - Camera, [130](#)
  - NotificationHelper, [76](#)
  - Player, [297](#)
- reset\_but\_label
  - MessageToPlayerWindow, [277](#)
- reset\_camera
  - Game, [196](#)
- reset\_research
  - ResearchWindow, [316](#)
- reset\_speed
  - MovementHelper, [73](#)
- reset\_timer
  - TriggerHelper, [106](#)
- reset\_unlocks
  - Game, [197](#)
- resolutions\_
  - OptionsWindow, [290](#)
- root\_
  - Game, [200](#)
- rotate
  - GraphicsHelper, [49](#)
- rows\_
  - ResearchWindow, [316](#)
- run
  - Game, [197](#)
- run\_away\_from
  - CombatSystem, [142](#)
- run\_away\_from\_
  - CombatSystem, [142](#)
- run\_away\_queue\_
  - CombatSystem, [143](#)
- SPELL\_TAB
  - action, [19](#)
- save\_component
  - GameSerializer, [207](#)
- save\_entities\_
  - GameSerializer, [208](#)
- save\_game
  - GameSerializer, [208](#)
- save\_tasks
  - GameSerializer, [208](#)
- save\_unlocks
  - GameSerializer, [208](#)
- save\_wave\_system
  - GameSerializer, [208](#)
- scene\_
  - EntitySystem, [176](#)
- scene\_mgr\_
  - GridSystem, [225](#)
  - SelectionBox, [331](#)
- Script
  - lpp::Script, [320](#)
- script\_
  - GameSerializer, [209](#)
  - MessageToPlayerWindow, [278](#)
  - ResearchWindow, [317](#)
  - SpellCastingWindow, [340](#)
  - Spellcaster, [336](#)
  - WaveSystem, [370](#)
- scroll\_down
  - Console, [147](#)
- second\_timer\_
  - WaveSystem, [370](#)
- select\_object
  - SelectionBox, [329](#)
- selected\_entities\_
  - SelectionBox, [331](#)
- selection\_box\_
  - Game, [200](#)
- selection\_in\_progress\_
  - SelectionBox, [331](#)
- selection\_number\_
  - BuilderWindow, [126](#)
  - SpellCastingWindow, [341](#)
- SelectionBox, [326](#)
  - ~SelectionBox, [327](#)
  - clear\_selected\_entities, [328](#)
  - entities\_, [330](#)
  - execute\_selection, [328](#)
  - execute\_single\_selection, [328](#)
  - extend\_to, [329](#)
  - get\_selected\_entities, [329](#)
  - is\_selecting, [329](#)
  - ray\_query\_, [330](#)
  - scene\_mgr\_, [331](#)
  - select\_object, [329](#)

- selected\_entities\_, 331
- selection\_in\_progress\_, 331
- SelectionBox, 327
- set\_corners, 329, 330
- set\_selecting, 330
- set\_starting\_point, 330
- start\_, 331
- volume\_query\_, 331
- selector\_
  - Spellcaster, 336
- serializers\_
  - GameSerializer, 209
- set
  - ExperienceValueHelper, 38
  - lpp::Script, 325
- set\_2d\_position
  - PhysicsHelper, 83
- set\_active
  - EventHelper, 35
- set\_alive
  - HealthHelper, 58
- set\_atk\_type
  - CombatHelper, 23
- set\_blueprint
  - AIHelper, 20
  - ConstructorHelper, 28
  - DestructorHelper, 31
  - OnHitHelper, 78
  - SpellHelper, 92
  - TaskHandlerHelper, 97
  - TriggerHelper, 106
  - UpgradeHelper, 110
- set\_butt\_label
  - MessageToPlayerWindow, 277
- set\_capacity
  - ManaCrystalHelper, 67
- set\_caster
  - SpellCastingWindow, 340
- set\_check\_period
  - TriggerSystem, 357
- set\_command
  - CommandHelper, 27
- set\_complete
  - TaskHelper, 100
- set\_component
  - EntitySystem, 175
- set\_cooldown
  - CombatHelper, 23
  - NotificationHelper, 77
  - OnHitHelper, 78
  - SpellHelper, 92
  - TriggerHelper, 107
- set\_corners
  - SelectionBox, 329, 330
- set\_countdown\_value
  - WaveSystem, 366
- set\_countdown\_window
  - WaveSystem, 366
- set\_curr\_gold
  - GoldHelper, 43
- set\_curr\_time
  - SpellHelper, 92
- set\_curr\_tool
  - GUI, 231
- set\_curr\_tool\_visible
  - GUI, 231
- set\_curr\_value
  - CounterHelper, 30
- set\_curr\_wave\_number
  - WaveSystem, 366
- set\_current\_entity\_table
  - EntityPlacer, 159
- set\_defense
  - HealthHelper, 58
- set\_delta
  - ExplosionHelper, 40
- set\_direction
  - Camera, 130
- set\_dmg
  - HomingHelper, 61
- set\_dmg\_range
  - CombatHelper, 24
- set\_endless\_mode
  - WaveSystem, 366
- set\_entities\_spawned
  - WaveSystem, 367
- set\_entity\_total
  - WaveSystem, 367
- set\_event\_handler
  - EventHelper, 36
- set\_event\_type
  - EventHelper, 36
- set\_exp\_needed
  - UpgradeHelper, 111
- set\_experience
  - UpgradeHelper, 111
- set\_faction
  - FactionHelper, 41
- set\_first\_person
  - InputSystem, 248
- set\_free
  - GridNodeHelper, 53
- set\_free\_mode
  - Camera, 130
- set\_free\_selected
  - GridNodeHelper, 53
- set\_half\_height
  - PhysicsHelper, 84
- set\_handler
  - EventHandlerHelper, 33
- set\_health
  - HealthHelper, 58
- set\_height
  - Camera, 131
- set\_history
  - Console, 148

- GameLog, 203
- set\_initial\_unlocks
  - Player, 297
- set\_input\_handler
  - InputHelper, 62
- set\_key\_bind
  - OptionsWindow, 289
- set\_last\_spell\_id
  - Spellcaster, 334
- set\_level
  - UpgradeHelper, 111
- set\_level\_cap
  - UpgradeHelper, 111
- set\_linked\_entity
  - TriggerHelper, 107
- set\_mana
  - ManaHelper, 70
- set\_manual\_scaling
  - GraphicsHelper, 49
- set\_material
  - GraphicsHelper, 49
- set\_max\_gold
  - GoldHelper, 44
- set\_max\_mana
  - ManaHelper, 70
- set\_max\_radius
  - ExplosionHelper, 40
- set\_max\_run\_away\_attempts
  - CombatSystem, 142
- set\_max\_time
  - LimitedLifeSpanHelper, 66
- set\_max\_value
  - CounterHelper, 30
- set\_mesh
  - GraphicsHelper, 50
- set\_name
  - NameHelper, 74
- set\_original\_speed
  - MovementHelper, 73
- set\_pathfinding\_blueprint
  - PathfindingHelper, 80
- set\_placer
  - BuilderWindow, 125
- set\_portal\_neighbour
  - GridNodeHelper, 54
- set\_position
  - Camera, 131
  - PhysicsHelper, 84
- set\_price
  - PriceHelper, 85
- set\_producer
  - ProductHelper, 86
- set\_production\_blueprint
  - ProductionHelper, 88
- set\_production\_cooldown
  - ProductionHelper, 89
- set\_production\_count
  - ProductionHelper, 89
- set\_production\_limit
  - ProductionHelper, 89
- set\_production\_progress
  - ProductionHelper, 89
- set\_projectile\_blueprint
  - CombatHelper, 24
- set\_query\_flags
  - GraphicsHelper, 50
- set\_radius
  - EventHelper, 36
  - StructureHelper, 94
  - TriggerHelper, 107
- set\_range
  - CombatHelper, 24
- set\_regen
  - HealthHelper, 58
  - ManaCrystalHelper, 68
  - ManaHelper, 70
- set\_regen\_period
  - HealthSystem, 242
  - ManaSpellSystem, 274
- set\_resident
  - GridNodeHelper, 54
- set\_scale
  - GraphicsHelper, 50
- set\_selecting
  - SelectionBox, 330
- set\_solid
  - PhysicsHelper, 84
- set\_source
  - HomingHelper, 61
- set\_spawn\_cooldown
  - WaveSystem, 367
- set\_spawn\_timer
  - WaveSystem, 367
- set\_speed\_modifier
  - MovementHelper, 74
- set\_spell
  - Spellcaster, 335
- set\_spell\_active
  - SpellCastingWindow, 340
- set\_spell\_type
  - Spellcaster, 335
- set\_start
  - Camera, 131
- set\_starting\_point
  - SelectionBox, 330
- set\_state
  - AIHelper, 20
  - Game, 197
  - WaveSystem, 367
- set\_target
  - CombatHelper, 25
  - EventHelper, 37
  - HomingHelper, 61
  - TimeHelper, 103
- set\_task\_source
  - TaskHelper, 100

- set\_task\_target
  - TaskHelper, 100
- set\_task\_type
  - TaskHelper, 101
- set\_throne\_id
  - Game, 197
- set\_time\_limit
  - TimeHelper, 104
- set\_time\_multiplier
  - ProductionSystem, 306
  - TimeSystem, 352
- set\_tracked\_entity
  - EntityTracker, 180
- set\_type
  - TimeHelper, 104
- set\_update\_period
  - AISystem, 120
  - EventSystem, 184
  - GraphicsSystem, 212
- set\_update\_time\_multiplier
  - EventSystem, 185
- set\_visible
  - Console, 148
  - EntityPlacer, 159
  - GUIWindow, 236
  - GUI, 231
  - GraphicsHelper, 50
  - GridSystem, 224
  - LightHelper, 64
- set\_walk\_through
  - StructureHelper, 94
- set\_wave\_count
  - WaveSystem, 368
- set\_wave\_entities
  - WaveSystem, 368
- set\_wave\_table
  - WaveSystem, 368
- show
  - MessageToPlayerWindow, 278
  - ResearchWindow, 316
- show\_load\_save\_dialog
  - GUI, 232
- show\_upgrade\_but
  - EntityTracker, 180
- spawn
  - WaveSystem, 368
- spawn\_entity
  - ProductionSystem, 306
- spawn\_timer\_
  - WaveSystem, 371
- spawning\_nodes\_
  - WaveSystem, 371
- speed\_
  - Camera, 132
- spell\_caster\_
  - Game, 201
- spell\_casting\_
  - GUI, 233
- SpellCastingWindow, 336
  - ~SpellCastingWindow, 338
  - cast, 338
  - caster\_, 340
  - clear\_spells, 338
  - curr\_active\_spell\_, 340
  - deactivate\_current\_spell, 338
  - dec\_selection, 338
  - get\_spell\_, 339
  - get\_spells, 339
  - inc\_selection, 339
  - init\_, 339
  - register\_spell, 339
  - script\_, 340
  - selection\_number\_, 341
  - set\_caster, 340
  - set\_spell\_active, 340
  - SpellCastingWindow, 338
  - spells\_, 341
  - update\_selection\_, 340
- SpellComponent, 341
- SpellHelper, 90
  - advance\_curr\_time, 90
  - cast, 91
  - get\_blueprint, 91
  - get\_cooldown, 91
  - get\_curr\_time, 91
  - set\_blueprint, 92
  - set\_cooldown, 92
  - set\_curr\_time, 92
- Spellcaster, 332
  - ~Spellcaster, 333
  - cast, 333
  - curr\_spell\_, 335
  - get\_last\_spell, 334
  - get\_last\_spell\_id, 334
  - get\_last\_spell\_type, 334
  - get\_spell, 334
  - get\_spell\_type, 334
  - is\_casting, 334
  - last\_spell\_, 335
  - last\_spell\_id\_, 335
  - placer\_, 336
  - script\_, 336
  - selector\_, 336
  - set\_last\_spell\_id, 334
  - set\_spell, 335
  - set\_spell\_type, 335
  - Spellcaster, 333
  - stop\_casting, 335
- Spellcaster::SPELL, 331
- spells\_
  - SpellCastingWindow, 341
- start
  - WaveSystem, 368
- start\_
  - Camera, 133
  - Grid, 220

- SelectionBox, 331
- start\_wave
  - WaveSystem, 369
- starting\_index\_
  - Grid, 220
- state\_
  - Game, 201
  - WaveSystem, 371
- stop\_casting
  - Spellcaster, 335
- structure\_radius\_
  - EntityPlacer, 161
- StructureComponent, 342
- StructureHelper, 93
  - add\_residence, 93
  - add\_residences, 93
  - get\_radius, 94
  - is\_walk\_through, 94
  - set\_radius, 94
  - set\_walk\_through, 94
- sub\_curr\_unit
  - Player, 298
- sub\_defense
  - HealthHelper, 59
- sub\_gold
  - GoldHelper, 44
  - Player, 298
- sub\_health
  - HealthHelper, 59
- sub\_mana
  - ManaHelper, 70
  - Player, 298
- sub\_mana\_regen
  - Player, 298
- sub\_max\_mana
  - Player, 299
- sub\_max\_unit
  - Player, 299
- System, 343
  - ~System, 344
  - update, 344
- systems\_
  - Game, 201
- table\_name\_
  - EntityPlacer, 162
- task\_backup\_
  - InputSystem, 249
- task\_names\_
  - TaskSystem, 349
- task\_pairs\_
  - GameSerializer, 209
- task\_possible
  - TaskHandlerHelper, 97
- TaskComponent, 344
- TaskHandlerComponent, 345
- TaskHandlerHelper, 95
  - add\_possible\_task, 95
  - clear\_task\_queue, 96
  - delete\_possible\_task, 96
  - get\_blueprint, 96
  - get\_task\_queue, 96
  - set\_blueprint, 97
  - task\_possible, 97
- TaskHelper, 97
  - add\_task, 98
  - cancel\_task, 98
  - create\_task, 99
  - get\_task\_source, 99
  - get\_task\_target, 99
  - get\_task\_type, 99
  - is\_complete, 100
  - set\_complete, 100
  - set\_task\_source, 100
  - set\_task\_target, 100
  - set\_task\_type, 101
- TaskSystem, 346
  - ~TaskSystem, 347
  - combat\_, 349
  - current\_task\_completed\_, 347
  - entities\_, 349
  - get\_task\_name, 348
  - grid\_, 349
  - handle\_task\_, 348
  - next\_task\_, 348
  - task\_names\_, 349
  - TaskSystem, 347
  - update, 348
- tdelta\_
  - TopBar, 355
- tdt, 101
- test\_command
  - CommandHelper, 27
- throne\_id\_
  - Game, 201
- time\_
  - util::effect::FREEZE\_EFFECT, 191
  - util::effect::LOWER\_SPEED\_EFFECT, 261
- time\_multiplier\_
  - ProductionSystem, 308
  - TimeSystem, 353
- time\_since\_last\_fps\_update\_
  - Console, 149
- TimeComponent, 349
- TimeHelper, 101
  - advance\_curr\_time, 102
  - get\_curr\_time, 102
  - get\_target, 103
  - get\_time\_limit, 103
  - get\_type, 103
  - max\_curr\_time, 103
  - set\_target, 103
  - set\_time\_limit, 104
  - set\_type, 104
- TimeSystem, 350
  - ~TimeSystem, 351
  - advance\_all\_timers, 351



- advance\_all\_timers\_of\_type, 351
- entities\_, 353
- get\_time\_multiplier, 352
- handle\_event\_, 352
- set\_time\_multiplier, 352
- time\_multiplier\_, 353
- TimeSystem, 351
- update, 352
- to\_be\_destroyed\_
  - EntitySystem, 177
- toggle\_camera\_free\_mode
  - Game, 197
- toggle\_placing\_when\_game\_paused
  - EntityPlacer, 160
- toggle\_visible
  - LightHelper, 65
- top\_bar\_
  - GUI, 233
- TopBar, 353
  - ~TopBar, 354
  - init\_, 354
  - tdelta\_, 355
  - TopBar, 354
  - update\_label, 354
  - update\_time, 355
- tracker\_
  - GUI, 233
- transfer\_all\_gold
  - GoldHelper, 44
- trigger
  - TriggerHelper, 107
- TriggerComponent, 355
- TriggerHelper, 104
  - can\_be\_triggered\_by, 105
  - get\_blueprint, 105
  - get\_cooldown, 106
  - get\_linked\_entity, 106
  - get\_radius, 106
  - reset\_timer, 106
  - set\_blueprint, 106
  - set\_cooldown, 107
  - set\_linked\_entity, 107
  - set\_radius, 107
  - trigger, 107
- TriggerSystem, 356
  - ~TriggerSystem, 357
  - check\_timer\_, 358
  - entities\_, 358
  - get\_check\_period, 357
  - set\_check\_period, 357
  - TriggerSystem, 357
  - update, 358
- ubercharge
  - HealthHelper, 59
- uint\_max\_
  - Player, 300
- units\_curr\_
  - Player, 300
- units\_max\_
  - Player, 300
- unlock
  - ResearchWindow, 316
- unlocked\_
  - ResearchWindow, 317
- update
  - AISystem, 120
  - Camera, 131
  - CombatSystem, 142
  - EntitySystem, 175
  - EventSystem, 185
  - Game, 197
  - GraphicsSystem, 212
  - GridSystem, 224
  - HealthSystem, 242
  - InputSystem, 248
  - ManaSpellSystem, 275
  - MovementSystem, 282
  - ProductionSystem, 306
  - System, 344
  - TaskSystem, 348
  - TimeSystem, 352
  - TriggerSystem, 358
  - WaveSystem, 369
- update\_fps
  - Console, 148
- update\_label
  - TopBar, 354
- update\_label\_text
  - WaveSystem, 369
- update\_labels\_
  - OptionsWindow, 289
- update\_neighbours\_
  - GridSystem, 224
- update\_period\_
  - EventSystem, 185
- update\_position
  - EntityPlacer, 160
- update\_regen
  - HealthSystem, 242
- update\_selection\_
  - BuilderWindow, 125
  - SpellCastingWindow, 340
- update\_time
  - TopBar, 355
- update\_time\_multiplier\_
  - EventSystem, 185
- update\_timer\_
  - AISystem, 120
  - GraphicsSystem, 212
- update\_tracking
  - EntityTracker, 180
- upgrade
  - UpgradeHelper, 112
- UpgradeComponent, 358
- UpgradeHelper, 108
  - add\_experience, 109

- can\_level\_up, 109
- get\_blueprint, 109
- get\_exp\_needed, 109
- get\_experience, 110
- get\_level, 110
- get\_level\_cap, 110
- set\_blueprint, 110
- set\_exp\_needed, 111
- set\_experience, 111
- set\_level, 111
- set\_level\_cap, 111
- upgrade, 112
- util, 112
  - abs, 113
  - DEFAULT\_PATH\_TYPE, 113
  - get\_enum\_direction, 114
  - get\_random, 114
  - pathfind, 114
- util::EntityDestroyer, 156
  - destroy, 156
- util::HAS\_GOLD, 237
  - ~HAS\_GOLD, 237
  - entities\_, 238
  - HAS\_GOLD, 237
  - operator(), 238
- util::IS\_ENEMY, 250
  - ~IS\_ENEMY, 250
  - enemy\_faction\_, 251
  - entities\_, 251
  - IS\_ENEMY, 250
  - operator(), 251
- util::IS\_FRIENDLY\_OR\_NEUTRAL, 253
  - ~IS\_FRIENDLY\_OR\_NEUTRAL, 254
  - entities\_, 254
  - faction\_, 254
  - IS\_FRIENDLY\_OR\_NEUTRAL, 253
  - operator(), 254
- util::IS\_FRIENDLY, 251
  - ~IS\_FRIENDLY, 252
  - entities\_, 252
  - faction\_, 253
  - IS\_FRIENDLY, 252
  - operator(), 252
- util::IS\_GOLD\_VAULT, 255
  - ~IS\_GOLD\_VAULT, 255
  - entities\_, 256
  - IS\_GOLD\_VAULT, 255
  - operator(), 256
- util::effect, 115
- util::effect::DAMAGE\_EFFECT, 151
  - ~DAMAGE\_EFFECT, 151
  - DAMAGE\_EFFECT, 151
  - entities\_, 152
  - min\_, 152
  - operator(), 152
- util::effect::FREEZE\_EFFECT, 190
  - ~FREEZE\_EFFECT, 191
  - entities\_, 191
- util::effect::HEAL\_EFFECT, 238
  - ~HEAL\_EFFECT, 239
  - entities\_, 239
  - HEAL\_EFFECT, 239
  - operator(), 239
- util::effect::LOWER\_SPEED\_EFFECT, 259
  - ~LOWER\_SPEED\_EFFECT, 260
  - entities\_, 260
  - LOWER\_SPEED\_EFFECT, 260
  - operator(), 260
  - time\_, 261
- util::heuristic, 115
- util::heuristic::HEURISTIC, 243
- util::heuristic::MANHATTAN\_DISTANCE, 275
- util::heuristic::NO\_HEURISTIC, 284
- util::heuristic::PORTAL\_HEURISTIC, 301
  - get\_closest\_portal, 301
- util::heuristic::RUN\_AWAY\_HEURISTIC, 317
- util::path\_type, 116
- util::path\_type::BEST\_PATH, 122
- util::path\_type::FIRST\_PATH, 190
- util::path\_type::RANDOM\_PATH< UPPER >, 308
- util::pathfinding, 116
- util::pathfinding::A\_STAR< PATH\_TYPE >, 117
- view\_
  - OptionsWindow, 290
- visible\_
  - EntityPlacer, 162
- volume\_query\_
  - SelectionBox, 331
- wave\_count\_
  - WaveSystem, 371
- wave\_entities\_
  - WaveSystem, 371
- wave\_entity\_died
  - WaveSystem, 369
- wave\_table\_
  - WaveSystem, 371
- WaveSystem, 359
  - ~WaveSystem, 363
  - add\_entity\_blueprint, 363
  - add\_spawn\_node, 363
  - advance\_wave\_countdown, 363
  - clear\_entity\_blueprints, 363
  - clear\_spawn\_nodes, 363
  - curr\_wave\_number\_, 369
  - end\_wave, 364
  - endless\_mode\_, 369
  - entities\_, 370
  - entities\_spawned\_, 370
  - entities\_total\_, 370
  - entity\_blueprints\_, 370
  - get\_countdown\_value, 364
  - get\_curr\_wave\_number, 364

- get\_endless\_mode, [364](#)
- get\_entities\_spawned, [364](#)
- get\_entity\_blueprints, [364](#)
- get\_entity\_total, [364](#)
- get\_spawn\_cooldown, [364](#)
- get\_spawn\_timer, [365](#)
- get\_spawning\_nodes, [365](#)
- get\_state, [365](#)
- get\_wave\_count, [365](#)
- get\_wave\_entities, [365](#)
- get\_wave\_table, [365](#)
- label\_text\_, [370](#)
- next\_wave, [365](#)
- next\_wave\_countdown\_, [370](#)
- pause, [365](#)
- script\_, [370](#)
- second\_timer\_, [370](#)
- set\_countdown\_value, [366](#)
- set\_countdown\_window, [366](#)
- set\_curr\_wave\_number, [366](#)
- set\_endless\_mode, [366](#)
- set\_entities\_spawned, [367](#)
- set\_entity\_total, [367](#)
- set\_spawn\_cooldown, [367](#)
- set\_spawn\_timer, [367](#)
- set\_state, [367](#)
- set\_wave\_count, [368](#)
- set\_wave\_entities, [368](#)
- set\_wave\_table, [368](#)
- spawn, [368](#)
- spawn\_timer\_, [371](#)
- spawning\_nodes\_, [371](#)
- start, [368](#)
- start\_wave, [369](#)
- state\_, [371](#)
- update, [369](#)
- update\_label\_text, [369](#)
- wave\_count\_, [371](#)
- wave\_entities\_, [371](#)
- wave\_entity\_died, [369](#)
- wave\_table\_, [371](#)
- WaveSystem, [362](#)
- window\_, [371](#)
- what
  - lpp::Exception, [187](#)
- what\_lua
  - lpp::Exception, [187](#)
- width\_
  - Grid, [220](#)
  - OptionsWindow, [290](#)
- window\_
  - GUIWindow, [236](#)
  - GUI, [233](#)
  - WaveSystem, [371](#)
- windowClosed
  - Game, [199](#)
- windowResized
  - Game, [199](#)
- yaw
  - Camera, [132](#)