

HologramDepthmap Library

Generated by Doxygen 1.8.13

Contents

1	Main Page	1
1.1	Introduction	1
1.2	Algorithm Reference	1
1.3	Software Components	2
1.4	Main Procedure	3
1.5	Environment	3
1.6	How to Build Source Codes	3
2	Module Index	5
2.1	Modules	5
3	Namespace Index	7
4	Hierarchical Index	9
4.1	Class Hierarchy	9
5	Class Index	11
5.1	Class List	11
6	File Index	13
6.1	File List	13

7	Module Documentation	15
7.1	Initialize	15
7.1.1	Detailed Description	15
7.1.2	Function Documentation	15
7.1.2.1	init_CPU()	15
7.1.2.2	init_GPU()	16
7.1.2.3	initialize()	16
7.1.2.4	readConfig()	16
7.2	Loading Data	17
7.2.1	Detailed Description	17
7.2.2	Function Documentation	17
7.2.2.1	prepare_inputdata_CPU()	17
7.2.2.2	prepare_inputdata_GPU()	17
7.2.2.3	ReadImageDepth()	18
7.3	Computing Depth Value	19
7.3.1	Detailed Description	19
7.3.2	Function Documentation	19
7.3.2.1	change_depth_quan_CPU()	19
7.3.2.2	change_depth_quan_GPU()	19
7.3.2.3	GetDepthValues()	19
7.4	Transform	20
7.4.1	Detailed Description	20
7.4.2	Function Documentation	20
7.4.2.1	TransformViewingWindow()	20
7.5	Generation Hologram	21
7.5.1	Detailed Description	21
7.5.2	Function Documentation	21
7.5.2.1	Calc_Holo_by_Depth()	21
7.5.2.2	Calc_Holo_CPU()	21
7.5.2.3	Calc_Holo_GPU()	22

7.5.2.4	GenerateHologram()	23
7.5.2.5	Propagation_AngularSpectrum_CPU()	23
7.5.2.6	Propagation_AngularSpectrum_GPU()	23
7.6	Encoding	26
7.6.1	Detailed Description	26
7.6.2	Function Documentation	26
7.6.2.1	encoding_CPU()	26
7.6.2.2	encoding_GPU()	26
7.6.2.3	Encoding_Symmetrization()	27
7.7	Writing Image	28
7.7.1	Detailed Description	28
7.7.2	Function Documentation	28
7.7.2.1	Write_Result_image()	28
7.8	Reconstruction	29
7.8.1	Detailed Description	29
7.8.2	Function Documentation	29
7.8.2.1	circshift()	29
7.8.2.2	ReconstructImage()	29
7.8.2.3	Reconstruction()	30
7.8.2.4	Test_Propagation_to_Eye_Pupil()	30
7.8.2.5	Write_Simulation_image()	30
7.9	GPU Modules	31
7.9.1	Detailed Description	31
7.9.2	Function Documentation	31
7.9.2.1	cudaChangeDepthQuanKernel()	31
7.9.2.2	cudaCropFringe()	32
7.9.2.3	cudaDepthHoloKernel()	33
7.9.2.4	cudaFFT()	34
7.9.2.5	cudaGetFringe()	34
7.9.2.6	cudaPropagation_AngularSpKernel()	35

8 Namespace Documentation	37
8.1 graphics Namespace Reference	37
8.1.1 Function Documentation	42
8.1.1.1 absolute() [1/3]	42
8.1.1.2 absolute() [2/3]	42
8.1.1.3 absolute() [3/3]	43
8.1.1.4 angle() [1/3]	43
8.1.1.5 angle() [2/3]	43
8.1.1.6 angle() [3/3]	43
8.1.1.7 apx_equal() [1/8]	43
8.1.1.8 apx_equal() [2/8]	44
8.1.1.9 apx_equal() [3/8]	44
8.1.1.10 apx_equal() [4/8]	44
8.1.1.11 apx_equal() [5/8]	44
8.1.1.12 apx_equal() [6/8]	44
8.1.1.13 apx_equal() [7/8]	45
8.1.1.14 apx_equal() [8/8]	45
8.1.1.15 cross()	45
8.1.1.16 inner() [1/3]	45
8.1.1.17 inner() [2/3]	45
8.1.1.18 inner() [3/3]	46
8.1.1.19 norm() [1/3]	46
8.1.1.20 norm() [2/3]	46
8.1.1.21 norm() [3/3]	46
8.1.1.22 operator!=(()) [1/3]	46
8.1.1.23 operator!=(()) [2/3]	47
8.1.1.24 operator!=(()) [3/3]	47
8.1.1.25 operator*() [1/18]	47
8.1.1.26 operator*() [2/18]	47
8.1.1.27 operator*() [3/18]	47

8.1.1.28	<code>operator*()</code> [4/18]	48
8.1.1.29	<code>operator*()</code> [5/18]	48
8.1.1.30	<code>operator*()</code> [6/18]	48
8.1.1.31	<code>operator*()</code> [7/18]	48
8.1.1.32	<code>operator*()</code> [8/18]	48
8.1.1.33	<code>operator*()</code> [9/18]	49
8.1.1.34	<code>operator*()</code> [10/18]	49
8.1.1.35	<code>operator*()</code> [11/18]	49
8.1.1.36	<code>operator*()</code> [12/18]	49
8.1.1.37	<code>operator*()</code> [13/18]	49
8.1.1.38	<code>operator*()</code> [14/18]	50
8.1.1.39	<code>operator*()</code> [15/18]	50
8.1.1.40	<code>operator*()</code> [16/18]	50
8.1.1.41	<code>operator*()</code> [17/18]	50
8.1.1.42	<code>operator*()</code> [18/18]	50
8.1.1.43	<code>operator*==()</code> [1/12]	51
8.1.1.44	<code>operator*==()</code> [2/12]	51
8.1.1.45	<code>operator*==()</code> [3/12]	51
8.1.1.46	<code>operator*==()</code> [4/12]	51
8.1.1.47	<code>operator*==()</code> [5/12]	51
8.1.1.48	<code>operator*==()</code> [6/12]	52
8.1.1.49	<code>operator*==()</code> [7/12]	52
8.1.1.50	<code>operator*==()</code> [8/12]	52
8.1.1.51	<code>operator*==()</code> [9/12]	52
8.1.1.52	<code>operator*==()</code> [10/12]	52
8.1.1.53	<code>operator*==()</code> [11/12]	53
8.1.1.54	<code>operator*==()</code> [12/12]	53
8.1.1.55	<code>operator+()</code> [1/18]	53
8.1.1.56	<code>operator+()</code> [2/18]	53
8.1.1.57	<code>operator+()</code> [3/18]	53

8.1.1.58	operator+()	[4/18]	54
8.1.1.59	operator+()	[5/18]	54
8.1.1.60	operator+()	[6/18]	54
8.1.1.61	operator+()	[7/18]	54
8.1.1.62	operator+()	[8/18]	54
8.1.1.63	operator+()	[9/18]	55
8.1.1.64	operator+()	[10/18]	55
8.1.1.65	operator+()	[11/18]	55
8.1.1.66	operator+()	[12/18]	55
8.1.1.67	operator+()	[13/18]	55
8.1.1.68	operator+()	[14/18]	56
8.1.1.69	operator+()	[15/18]	56
8.1.1.70	operator+()	[16/18]	56
8.1.1.71	operator+()	[17/18]	56
8.1.1.72	operator+()	[18/18]	56
8.1.1.73	operator+=()	[1/12]	57
8.1.1.74	operator+=()	[2/12]	57
8.1.1.75	operator+=()	[3/12]	57
8.1.1.76	operator+=()	[4/12]	57
8.1.1.77	operator+=()	[5/12]	57
8.1.1.78	operator+=()	[6/12]	58
8.1.1.79	operator+=()	[7/12]	58
8.1.1.80	operator+=()	[8/12]	58
8.1.1.81	operator+=()	[9/12]	58
8.1.1.82	operator+=()	[10/12]	58
8.1.1.83	operator+=()	[11/12]	59
8.1.1.84	operator+=()	[12/12]	59
8.1.1.85	operator-()	[1/24]	59
8.1.1.86	operator-()	[2/24]	59
8.1.1.87	operator-()	[3/24]	59

8.1.1.88 operator-() [4/24]	60
8.1.1.89 operator-() [5/24]	60
8.1.1.90 operator-() [6/24]	60
8.1.1.91 operator-() [7/24]	60
8.1.1.92 operator-() [8/24]	60
8.1.1.93 operator-() [9/24]	61
8.1.1.94 operator-() [10/24]	61
8.1.1.95 operator-() [11/24]	61
8.1.1.96 operator-() [12/24]	61
8.1.1.97 operator-() [13/24]	61
8.1.1.98 operator-() [14/24]	62
8.1.1.99 operator-() [15/24]	62
8.1.1.100 operator-() [16/24]	62
8.1.1.101 operator-() [17/24]	62
8.1.1.102 operator-() [18/24]	62
8.1.1.103 operator-() [19/24]	63
8.1.1.104 operator-() [20/24]	63
8.1.1.105 operator-() [21/24]	63
8.1.1.106 operator-() [22/24]	63
8.1.1.107 operator-() [23/24]	63
8.1.1.108 operator-() [24/24]	64
8.1.1.109 operator-==() [1/12]	64
8.1.1.110 operator-==() [2/12]	64
8.1.1.111 operator-==() [3/12]	64
8.1.1.112 operator-==() [4/12]	64
8.1.1.113 operator-==() [5/12]	65
8.1.1.114 operator-==() [6/12]	65
8.1.1.115 operator-==() [7/12]	65
8.1.1.116 operator-==() [8/12]	65
8.1.1.117 operator-==() [9/12]	65

8.1.1.118 operator==() [10/12]	66
8.1.1.119 operator==() [11/12]	66
8.1.1.120 operator==() [12/12]	66
8.1.1.121 operator/() [1/9]	66
8.1.1.122 operator/() [2/9]	66
8.1.1.123 operator/() [3/9]	67
8.1.1.124 operator/() [4/9]	67
8.1.1.125 operator/() [5/9]	67
8.1.1.126 operator/() [6/9]	67
8.1.1.127 operator/() [7/9]	67
8.1.1.128 operator/() [8/9]	68
8.1.1.129 operator/() [9/9]	68
8.1.1.130 operator/=() [1/6]	68
8.1.1.131 operator/=() [2/6]	68
8.1.1.132 operator/=() [3/6]	68
8.1.1.133 operator/=() [4/6]	69
8.1.1.134 operator/=() [5/6]	69
8.1.1.135 operator/=() [6/6]	69
8.1.1.136 operator<() [1/18]	69
8.1.1.137 operator<() [2/18]	69
8.1.1.138 operator<() [3/18]	70
8.1.1.139 operator<() [4/18]	70
8.1.1.140 operator<() [5/18]	70
8.1.1.141 operator<() [6/18]	70
8.1.1.142 operator<() [7/18]	70
8.1.1.143 operator<() [8/18]	71
8.1.1.144 operator<() [9/18]	71
8.1.1.145 operator<() [10/18]	71
8.1.1.146 operator<() [11/18]	71
8.1.1.147 operator<() [12/18]	71

8.1.1.148 operator<() [13/18]	72
8.1.1.149 operator<() [14/18]	72
8.1.1.150 operator<() [15/18]	72
8.1.1.151 operator<() [16/18]	72
8.1.1.152 operator<() [17/18]	72
8.1.1.153 operator<() [18/18]	73
8.1.1.154 operator<=() [1/18]	73
8.1.1.155 operator<=() [2/18]	73
8.1.1.156 operator<=() [3/18]	73
8.1.1.157 operator<=() [4/18]	73
8.1.1.158 operator<=() [5/18]	74
8.1.1.159 operator<=() [6/18]	74
8.1.1.160 operator<=() [7/18]	74
8.1.1.161 operator<=() [8/18]	74
8.1.1.162 operator<=() [9/18]	74
8.1.1.163 operator<=() [10/18]	75
8.1.1.164 operator<=() [11/18]	75
8.1.1.165 operator<=() [12/18]	75
8.1.1.166 operator<=() [13/18]	75
8.1.1.167 operator<=() [14/18]	75
8.1.1.168 operator<=() [15/18]	76
8.1.1.169 operator<=() [16/18]	76
8.1.1.170 operator<=() [17/18]	76
8.1.1.171 operator<=() [18/18]	76
8.1.1.172 operator==() [1/18]	76
8.1.1.173 operator==() [2/18]	77
8.1.1.174 operator==() [3/18]	77
8.1.1.175 operator==() [4/18]	77
8.1.1.176 operator==() [5/18]	77
8.1.1.177 operator==() [6/18]	77

8.1.1.178 operator==() [7/18]	78
8.1.1.179 operator==() [8/18]	78
8.1.1.180 operator==() [9/18]	78
8.1.1.181 operator==() [10/18]	78
8.1.1.182 operator==() [11/18]	78
8.1.1.183 operator==() [12/18]	79
8.1.1.184 operator==() [13/18]	79
8.1.1.185 operator==() [14/18]	79
8.1.1.186 operator==() [15/18]	79
8.1.1.187 operator==() [16/18]	79
8.1.1.188 operator==() [17/18]	80
8.1.1.189 operator==() [18/18]	80
8.1.1.190 operator>() [1/18]	80
8.1.1.191 operator>() [2/18]	80
8.1.1.192 operator>() [3/18]	80
8.1.1.193 operator>() [4/18]	81
8.1.1.194 operator>() [5/18]	81
8.1.1.195 operator>() [6/18]	81
8.1.1.196 operator>() [7/18]	81
8.1.1.197 operator>() [8/18]	81
8.1.1.198 operator>() [9/18]	82
8.1.1.199 operator>() [10/18]	82
8.1.1.200 operator>() [11/18]	82
8.1.1.201 operator>() [12/18]	82
8.1.1.202 operator>() [13/18]	82
8.1.1.203 operator>() [14/18]	83
8.1.1.204 operator>() [15/18]	83
8.1.1.205 operator>() [16/18]	83
8.1.1.206 operator>() [17/18]	83
8.1.1.207 operator>() [18/18]	83

8.1.1.208 operator>=()	[1/18]	84
8.1.1.209 operator>=()	[2/18]	84
8.1.1.210 operator>=()	[3/18]	84
8.1.1.211 operator>=()	[4/18]	84
8.1.1.212 operator>=()	[5/18]	84
8.1.1.213 operator>=()	[6/18]	85
8.1.1.214 operator>=()	[7/18]	85
8.1.1.215 operator>=()	[8/18]	85
8.1.1.216 operator>=()	[9/18]	85
8.1.1.217 operator>=()	[10/18]	85
8.1.1.218 operator>=()	[11/18]	86
8.1.1.219 operator>=()	[12/18]	86
8.1.1.220 operator>=()	[13/18]	86
8.1.1.221 operator>=()	[14/18]	86
8.1.1.222 operator>=()	[15/18]	86
8.1.1.223 operator>=()	[16/18]	87
8.1.1.224 operator>=()	[17/18]	87
8.1.1.225 operator>=()	[18/18]	87
8.1.1.226 proj()	[1/3]	87
8.1.1.227 proj()	[2/3]	87
8.1.1.228 proj()	[3/3]	88
8.1.1.229 reset_u_epsilon()		88
8.1.1.230 reset_zero_epsilon()		88
8.1.1.231 scan()	[1/3]	88
8.1.1.232 scan()	[2/3]	88
8.1.1.233 scan()	[3/3]	89
8.1.1.234 set_u_epsilon()		89
8.1.1.235 set_zero_epsilon()		89
8.1.1.236 squaredNorm()	[1/3]	89
8.1.1.237 squaredNorm()	[2/3]	89

8.1.1.238 squaredNorm() [3/3]	90
8.1.1.239 store() [1/3]	90
8.1.1.240 store() [2/3]	90
8.1.1.241 store() [3/3]	90
8.1.1.242 sum() [1/3]	90
8.1.1.243 sum() [2/3]	91
8.1.1.244 sum() [3/3]	91
8.1.1.245 unit() [1/3]	91
8.1.1.246 unit() [2/3]	91
8.1.1.247 unit() [3/3]	91
8.1.2 Variable Documentation	91
8.1.2.1 angle_tolerance	92
8.1.2.2 epsilon	92
8.1.2.3 intersection_epsilon	92
8.1.2.4 save_zero_epsilon	92
8.1.2.5 sqrt_epsilon	92
8.1.2.6 unset_value	92
8.1.2.7 user_epsilon	93
8.1.2.8 zero_epsilon	93
8.1.2.9 zero_tolerance	93

9 Class Documentation	95
9.1 Complex Class Reference	95
9.1.1 Detailed Description	96
9.1.2 Constructor & Destructor Documentation	96
9.1.2.1 Complex() [1/3]	96
9.1.2.2 Complex() [2/3]	96
9.1.2.3 Complex() [3/3]	96
9.1.3 Member Function Documentation	96
9.1.3.1 arg()	96
9.1.3.2 conj()	97
9.1.3.3 euler()	97
9.1.3.4 mag()	97
9.1.3.5 mag2()	97
9.1.3.6 operator*=() [1/2]	97
9.1.3.7 operator*=() [2/2]	97
9.1.3.8 operator+=()	98
9.1.3.9 operator-=()	98
9.1.3.10 operator/=()	98
9.1.3.11 operator=()	98
9.1.4 Friends And Related Function Documentation	98
9.1.4.1 operator* [1/3]	98
9.1.4.2 operator* [2/3]	99
9.1.4.3 operator* [3/3]	99
9.1.4.4 operator+	99
9.1.4.5 operator-	99
9.1.4.6 operator/	99
9.1.4.7 operator<<	100
9.1.5 Member Data Documentation	100
9.1.5.1 a	100
9.1.5.2 b	100

9.2	HologramDepthmap Class Reference	100
9.2.1	Detailed Description	101
9.2.2	Constructor & Destructor Documentation	101
9.2.2.1	HologramDepthmap()	101
9.2.2.2	~HologramDepthmap()	101
9.2.3	Member Function Documentation	101
9.2.3.1	GenHologram	101
9.2.3.2	ReconImage	101
9.2.4	Member Data Documentation	102
9.2.4.1	hologram_	102
9.2.4.2	ui	102
9.3	HologramGenerator Class Reference	102
9.3.1	Detailed Description	106
9.3.2	Constructor & Destructor Documentation	107
9.3.2.1	HologramGenerator()	107
9.3.2.2	~HologramGenerator()	107
9.3.3	Member Function Documentation	107
9.3.3.1	exponent_complex()	107
9.3.3.2	fftShift()	107
9.3.3.3	fftwShift()	108
9.3.3.4	get_rand_phase_value()	109
9.3.3.5	get_shift_phase_value()	109
9.3.3.6	setMode()	109
9.3.4	Member Data Documentation	110
9.3.4.1	alpha_map_	110
9.3.4.2	DEFAULT_DEPTH_QUANTIZATION	110
9.3.4.3	depth_index_	110
9.3.4.4	depth_index_gpu_	110
9.3.4.5	DEPTH_PREFIX	110
9.3.4.6	dimg_src_gpu_	111

9.3.4.7	dlevel_	111
9.3.4.8	dlevel_transform_	111
9.3.4.9	dmap_	111
9.3.4.10	dmap_src_	111
9.3.4.11	dstep_	111
9.3.4.12	Encoding_Method_	112
9.3.4.13	eye_center_xy_	112
9.3.4.14	eye_length_	112
9.3.4.15	eye_pupil_diameter_	112
9.3.4.16	f_field_	112
9.3.4.17	FLAG_CHANGE_DEPTH_QUANTIZATION	112
9.3.4.18	FLAG_STATIC_IMAGE	113
9.3.4.19	focus_distance_	113
9.3.4.20	hh_complex_	113
9.3.4.21	IMAGE_PREFIX	113
9.3.4.22	img_src_	113
9.3.4.23	img_src_gpu_	113
9.3.4.24	isCPU_	114
9.3.4.25	NUMBER_OF_DEPTH_QUANTIZATION	114
9.3.4.26	NUMBER_OF_DIGIT_OF_FRAME_NUMBERING	114
9.3.4.27	NUMBER_OF_FRAME	114
9.3.4.28	params_	114
9.3.4.29	Pixel_pitch_xy_	114
9.3.4.30	Propagation_Method_	115
9.3.4.31	RANDOM_PHASE	115
9.3.4.32	RESULT_FOLDER	115
9.3.4.33	RESULT_PREFIX	115
9.3.4.34	sim_final_	115
9.3.4.35	sim_from_	115
9.3.4.36	sim_step_num_	116

9.3.4.37	<code>sim_to_</code>	116
9.3.4.38	<code>sim_type_</code>	116
9.3.4.39	<code>Simulation_Result_File_Prefix_</code>	116
9.3.4.40	<code>SLM_pixel_number_xy_</code>	116
9.3.4.41	<code>SOURCE_FOLDER</code>	116
9.3.4.42	<code>START_OF_FRAME_NUMBERING</code>	117
9.3.4.43	<code>test_pixel_number_scale_</code>	117
9.3.4.44	<code>Transform_Method_</code>	117
9.3.4.45	<code>u255_fringe_</code>	117
9.3.4.46	<code>U_complex_</code>	117
9.3.4.47	<code>WAVELENGTH</code>	117
9.4	HologramParams Struct Reference	118
9.4.1	Detailed Description	118
9.4.2	Member Data Documentation	118
9.4.2.1	<code>far_depthmap</code>	118
9.4.2.2	<code>field_lens</code>	119
9.4.2.3	<code>k</code>	119
9.4.2.4	<code>lambda</code>	119
9.4.2.5	<code>near_depthmap</code>	119
9.4.2.6	<code>num_of_depth</code>	119
9.4.2.7	<code>pn</code>	120
9.4.2.8	<code>pp</code>	120
9.4.2.9	<code>render_depth</code>	120
9.4.2.10	<code>ss</code>	120
9.5	graphics::ivec2 Struct Reference	120
9.5.1	Detailed Description	121
9.5.2	Constructor & Destructor Documentation	121
9.5.2.1	<code>ivec2()</code> [1/4]	121
9.5.2.2	<code>ivec2()</code> [2/4]	121
9.5.2.3	<code>ivec2()</code> [3/4]	121

9.5.2.4	ivec2() [4/4]	122
9.5.3	Member Function Documentation	122
9.5.3.1	operator>() [1/2]	122
9.5.3.2	operator>() [2/2]	122
9.5.3.3	operator=()	122
9.5.3.4	operator[]() [1/2]	122
9.5.3.5	operator[]() [2/2]	123
9.5.4	Member Data Documentation	123
9.5.4.1	n	123
9.5.4.2	v	123
9.6	graphics::ivec3 Struct Reference	123
9.6.1	Detailed Description	124
9.6.2	Constructor & Destructor Documentation	124
9.6.2.1	ivec3() [1/4]	124
9.6.2.2	ivec3() [2/4]	124
9.6.2.3	ivec3() [3/4]	124
9.6.2.4	ivec3() [4/4]	125
9.6.3	Member Function Documentation	125
9.6.3.1	operator>() [1/2]	125
9.6.3.2	operator>() [2/2]	125
9.6.3.3	operator=()	125
9.6.3.4	operator[]() [1/2]	125
9.6.3.5	operator[]() [2/2]	126
9.6.4	Member Data Documentation	126
9.6.4.1	n	126
9.6.4.2	v	126
9.7	graphics::ivec4 Struct Reference	126
9.7.1	Detailed Description	127
9.7.2	Constructor & Destructor Documentation	127
9.7.2.1	ivec4() [1/4]	127

9.7.2.2	ivec4() [2/4]	127
9.7.2.3	ivec4() [3/4]	127
9.7.2.4	ivec4() [4/4]	128
9.7.3	Member Function Documentation	128
9.7.3.1	operator>() [1/2]	128
9.7.3.2	operator>() [2/2]	128
9.7.3.3	operator=()	128
9.7.3.4	operator[]() [1/2]	128
9.7.3.5	operator[]() [2/2]	129
9.7.4	Member Data Documentation	129
9.7.4.1	n	129
9.7.4.2	v	129
9.8	graphics::vec2 Struct Reference	129
9.8.1	Detailed Description	130
9.8.2	Constructor & Destructor Documentation	130
9.8.2.1	vec2() [1/5]	130
9.8.2.2	vec2() [2/5]	130
9.8.2.3	vec2() [3/5]	130
9.8.2.4	vec2() [4/5]	131
9.8.2.5	vec2() [5/5]	131
9.8.3	Member Function Documentation	131
9.8.3.1	is_parallel()	131
9.8.3.2	is_perpendicular()	131
9.8.3.3	is_tiny()	131
9.8.3.4	is_zero()	132
9.8.3.5	length()	132
9.8.3.6	operator>() [1/2]	132
9.8.3.7	operator>() [2/2]	132
9.8.3.8	operator=()	132
9.8.3.9	operator[]() [1/2]	132

9.8.3.10	operator[]() [2/2]	133
9.8.3.11	perpendicular() [1/2]	133
9.8.3.12	perpendicular() [2/2]	133
9.8.3.13	unit()	133
9.8.4	Member Data Documentation	133
9.8.4.1	n	133
9.8.4.2	v	134
9.9	graphics::vec3 Struct Reference	134
9.9.1	Detailed Description	135
9.9.2	Constructor & Destructor Documentation	135
9.9.2.1	vec3() [1/5]	135
9.9.2.2	vec3() [2/5]	135
9.9.2.3	vec3() [3/5]	135
9.9.2.4	vec3() [4/5]	135
9.9.2.5	vec3() [5/5]	136
9.9.3	Member Function Documentation	136
9.9.3.1	is_parallel()	136
9.9.3.2	is_perpendicular()	136
9.9.3.3	is_tiny()	136
9.9.3.4	is_zero()	136
9.9.3.5	length()	137
9.9.3.6	operator()() [1/2]	137
9.9.3.7	operator()() [2/2]	137
9.9.3.8	operator=()	137
9.9.3.9	operator[]() [1/2]	137
9.9.3.10	operator[]() [2/2]	137
9.9.3.11	perpendicular() [1/2]	138
9.9.3.12	perpendicular() [2/2]	138
9.9.3.13	unit()	138
9.9.4	Member Data Documentation	138

9.9.4.1	n	138
9.9.4.2	v	138
9.10	graphics::vec4 Struct Reference	139
9.10.1	Detailed Description	139
9.10.2	Constructor & Destructor Documentation	139
9.10.2.1	vec4() [1/5]	139
9.10.2.2	vec4() [2/5]	140
9.10.2.3	vec4() [3/5]	140
9.10.2.4	vec4() [4/5]	140
9.10.2.5	vec4() [5/5]	140
9.10.3	Member Function Documentation	140
9.10.3.1	is_tiny()	140
9.10.3.2	is_zero()	141
9.10.3.3	length()	141
9.10.3.4	operator>() [1/2]	141
9.10.3.5	operator>() [2/2]	141
9.10.3.6	operator=()	141
9.10.3.7	operator[]() [1/2]	141
9.10.3.8	operator[]() [2/2]	142
9.10.3.9	unit()	142
9.10.4	Member Data Documentation	142
9.10.4.1	n	142
9.10.4.2	v	142

10 File Documentation	143
10.1 graphics/complex.h File Reference	143
10.1.1 Variable Documentation	143
10.1.1.1 PI	143
10.1.1.2 TWO_PI	143
10.2 graphics/epsilon.h File Reference	144
10.3 graphics/ivec.h File Reference	144
10.4 graphics/real.h File Reference	146
10.4.1 Macro Definition Documentation	147
10.4.1.1 _MAXDOUBLE	147
10.4.1.2 _MAXFLOAT	147
10.4.1.3 _MINDOUBLE	147
10.4.1.4 _MINFLOAT	148
10.4.1.5 M_PI	148
10.4.1.6 MAXREAL	148
10.4.1.7 MINREAL	148
10.4.1.8 REAL_T_IS_FLOAT	148
10.4.2 Typedef Documentation	148
10.4.2.1 real	148
10.4.2.2 real_t	149
10.5 graphics/src/epsilon.cpp File Reference	149
10.6 graphics/src/sys.cpp File Reference	149
10.6.1 Function Documentation	150
10.6.1.1 file_log()	150
10.6.1.2 file_read_open() [1/2]	150
10.6.1.3 file_read_open() [2/2]	150
10.6.1.4 file_write_open() [1/2]	150
10.6.1.5 file_write_open() [2/2]	151
10.6.1.6 string_cat()	151
10.6.1.7 string_cmp()	151

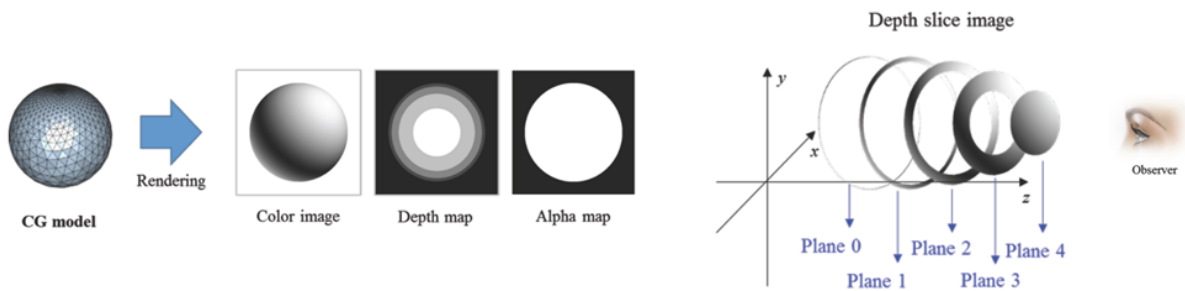
10.6.1.8	string_cpy()	151
10.6.2	Variable Documentation	151
10.6.2.1	fp	151
10.7	graphics/src/vec.cpp File Reference	152
10.8	graphics/sys.h File Reference	152
10.8.1	Macro Definition Documentation	153
10.8.1.1	FLOG	153
10.8.2	Function Documentation	153
10.8.2.1	file_read_open() [1/2]	153
10.8.2.2	file_read_open() [2/2]	153
10.8.2.3	file_write_open() [1/2]	153
10.8.2.4	file_write_open() [2/2]	153
10.8.2.5	string_cat()	154
10.8.2.6	string_cmp()	154
10.8.2.7	string_cpy()	154
10.9	graphics/vec.h File Reference	154
10.10	Hologram/HologramGenerator.h File Reference	157
10.11	Hologram/src/HologramGenerator.cpp File Reference	158
10.12	Hologram/src/HologramGenerator_CPU.cpp File Reference	158
10.12.1	Variable Documentation	158
10.12.1.1	fft_plan_bwd_	158
10.12.1.2	fft_plan_fwd_	158
10.13	Hologram/src/HologramGenerator_GPU.cpp File Reference	159
10.13.1	Macro Definition Documentation	160
10.13.1.1	HANDLE_ERROR	160
10.13.1.2	HANDLE_NULL	160
10.13.2	Function Documentation	160
10.13.2.1	HandleError()	160
10.13.3	Variable Documentation	160
10.13.3.1	k_temp_d_	161
10.13.3.2	start	161
10.13.3.3	stop	161
10.13.3.4	stream_	161
10.13.3.5	u_complex_gpu_	161
10.13.3.6	u_o_gpu_	161
10.14	Hologram/src/HologramKernel.cu File Reference	162
10.15	HologramDepthmap/hologramdepthmap.cpp File Reference	162
10.16	HologramDepthmap/hologramdepthmap.h File Reference	162
10.17	HologramDepthmap/main.cpp File Reference	162
10.17.1	Function Documentation	162
10.17.1.1	main()	162

Chapter 1

Main Page

1.1 Introduction

This library implements the hologram generation method using depth map data. It is implemented on the CPU and the GPU to improve the performance of the hologram generation method. Thus, user can compare the performance between the CPU and GPU implementation.



1.2 Algorithm Reference

The original algorithm is modified in the way that can be easily implemented in parallel. Back propagate each depth plane to the hologram plane and accumulate the results of each propagation.

1.4 Main Procedure

The main function of the library is a **GenerateHologram()** of **HologramGenerator** class. The following is the procedure of it and functions called from it..

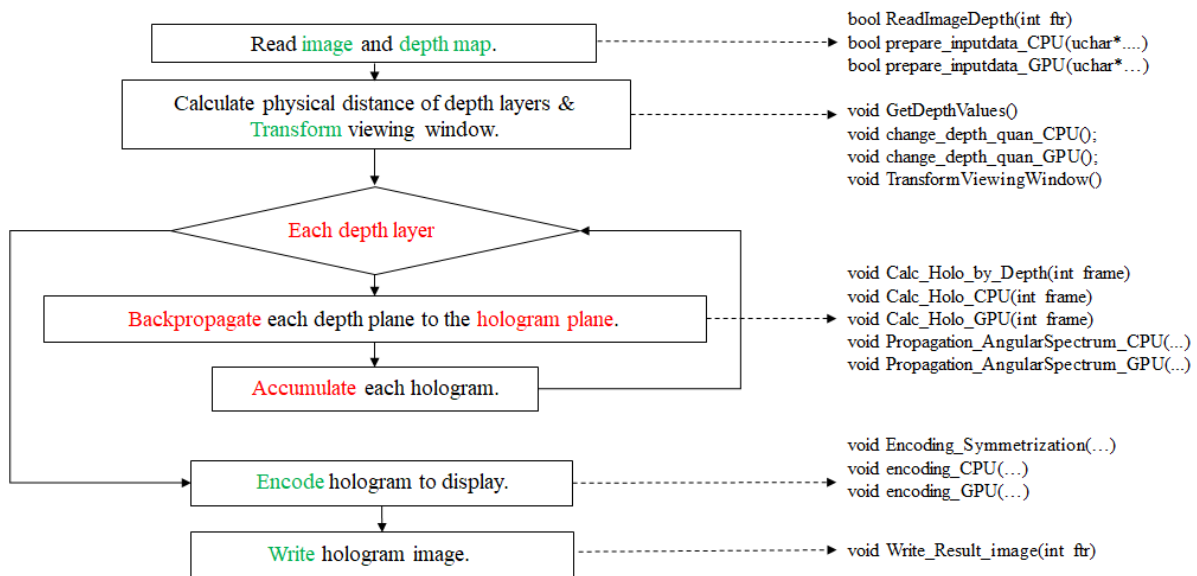


Figure 1.2 GenerateHologram Function Procedure

1.5 Environment

- Microsoft Visual Studio 2015 C++
- Qt 5.6.2
- CUDA 8.0
- FFTW 3.3.5

1.6 How to Build Source Codes

Before building an execution file, you need to install MS Visual Studio 2015 C++ and Qt, also CUDA for the GPU execution.

1. Download the source code from [here](#).
2. Go to the directory '**HologramDepthmap**'.
3. Open the Visual Studio solution file, '**HologramDepthmap.sln**'.
4. Check the configuration of the Qt & CUDA to work with the Visual Studio.
5. For Qt, you may need to set QTDIR environment variable -> System Properties->Advanced->Environment Variable.

6. To use FFTW, copy 'libfftw3-3.dll' into the 'bin' directory and copy 'libfftw3-3.lib' into the 'lib' directory.
7. Visual Studio Build Menu -> Configuration Menu, set "Release" for the Active solution configuration, "x64" for the Active solution platform.
8. Set '[HologramDepthmap](#)' as a StartUp Project.
9. Build a Solution.
10. After building, you can find the execution file, 'HologramDepthmap.exe' under the 'bin' directory.
11. Execute 'HologramDepthmap.exe', then you can see the following GUI of the sample program.

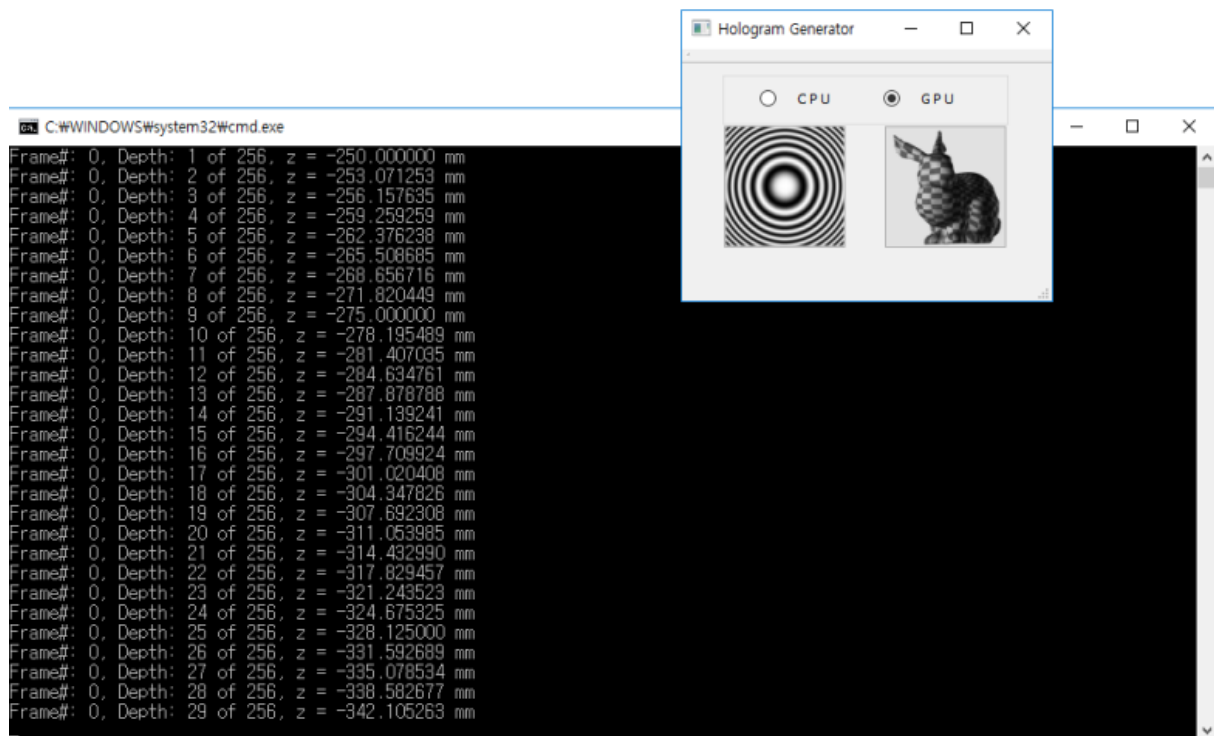


Figure 1.3 the Sample Program & its Execution

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Initialize	15
Loading Data	17
Computing Depth Value	19
Transform	20
Generation Hologram	21
Encoding	26
Writing Image	28
Reconstruction	29
GPU Modules	31

Chapter 3

Namespace Index

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Complex	95
HologramGenerator	102
HologramParams	118
graphics::ivec2	120
graphics::ivec3	123
graphics::ivec4	126
QMainWindow	
HologramDepthmap	100
graphics::vec2	129
graphics::vec3	134
graphics::vec4	139

Chapter 5

Class Index

5.1 Class List

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

Complex	Class for the complex number and its arithmetic	95
HologramDepthmap	Test class for executing the sample program, which shows how to use a hologram library . . .	100
HologramGenerator	Main class for generating a hologram using depth map data	102
HologramParams	Structure variable for hologram paramemters	118
graphics::ivec2	Structure for 2-dimensional integer vector and its arithmetic	120
graphics::ivec3	Structure for 3-dimensional integer vector and its arithmetic	123
graphics::ivec4	Structure for 4-dimensional integer vector and its arithmetic	126
graphics::vec2	Structure for 2-dimensional real type vector and its arithmetic	129
graphics::vec3	Structure for 3-dimensional real type vector and its arithmetic	134
graphics::vec4	Structure for 4-dimensional real type vector and its arithmetic	139

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

graphics/complex.h	143
graphics/epsilon.h	144
graphics/ivec.h	144
graphics/real.h	146
graphics/sys.h	152
graphics/vec.h	154
graphics/src/epsilon.cpp	149
graphics/src/sys.cpp	149
graphics/src/vec.cpp	152
Hologram/HologramGenerator.h	157
Hologram/src/HologramGenerator.cpp	158
Hologram/src/HologramGenerator_CPU.cpp	158
Hologram/src/HologramGenerator_GPU.cpp	159
Hologram/src/HologramKernel.cu	162
HologramDepthmap/hologramdepthmap.cpp	162
HologramDepthmap/hologramdepthmap.h	162
HologramDepthmap/main.cpp	162

Chapter 7

Module Documentation

7.1 Initialize

Functions

- bool [HologramGenerator::readConfig](#) ()
Read parameters from a config file(config_openholo.txt).
- void [HologramGenerator::initialize](#) ()
Initialize variables for CPU and GPU implementation.
- void [HologramGenerator::init_CPU](#) ()
Initialize variables for the CPU implementation.
- void [HologramGenerator::init_GPU](#) ()
Initialize variables for the GPU implementation.

7.1.1 Detailed Description

7.1.2 Function Documentation

7.1.2.1 [init_CPU\(\)](#)

```
void HologramGenerator::init_CPU ( ) [private]
```

Memory allocation for the CPU variables.

See also

[initialize](#)

Definition at line 13 of file HologramGenerator_CPU.cpp.

7.1.2.2 init_GPU()

```
void HologramGenerator::init_GPU ( ) [private]
```

Memory allocation for the GPU variables.

See also

[initialize](#)

Definition at line 157 of file HologramGenerator_GPU.cpp.

7.1.2.3 initialize()

```
void HologramGenerator::initialize ( )
```

See also

[init_CPU](#), [init_GPU](#)

Definition at line 235 of file HologramGenerator.cpp.

7.1.2.4 readConfig()

```
bool HologramGenerator::readConfig ( )
```

Returns

true if config information are successfully read, false otherwise.

Definition at line 65 of file HologramGenerator.cpp.

7.2 Loading Data

- bool [HologramGenerator::ReadImageDepth](#) (int ftr)
Read image and depth map.
- bool [HologramGenerator::prepare_inputdata_CPU](#) (uchar *img, uchar *dimg)
Preprocess input image & depth map data for the CPU implementation.
- bool [HologramGenerator::prepare_inputdata_GPU](#) (uchar *img, uchar *dimg)
Copy input image & depth map data into a GPU.

7.2.1 Detailed Description

7.2.2 Function Documentation

7.2.2.1 [prepare_inputdata_CPU\(\)](#)

```
bool HologramGenerator::prepare_inputdata_CPU (
    uchar * imgptr,
    uchar * dimgptr ) [private]
```

Prepare variables, `img_src_`, `dmap_src_`, `alpha_map_`, `depth_index_`.

Parameters

<i>imgptr</i>	: input image data pointer
<i>dimgptr</i>	: input depth map data pointer

Returns

true if input data are successfully prepared, false otherwise.

See also

[ReadImageDepth](#)

Definition at line 44 of file `HologramGenerator_CPU.cpp`.

7.2.2.2 [prepare_inputdata_GPU\(\)](#)

```
bool HologramGenerator::prepare_inputdata_GPU (
    uchar * imgptr,
    uchar * dimgptr ) [private]
```

Parameters

<i>imgptr</i>	: input image data pointer
<i>dimgptr</i>	: input depth map data pointer

Returns

true if input data are sucessfully copied on GPU, flase otherwise.

See also

[ReadImageDepth](#)

Definition at line 194 of file HologramGenerator_GPU.cpp.

7.2.2.3 ReadImageDepth()

```
bool HologramGenerator::ReadImageDepth (  
    int ftr ) [private]
```

Read input files and load image & depth map data. If the input image size is different with the dislay resolution, resize the image size.

Parameters

<i>ftr</i>	: the frame number of the image.
------------	----------------------------------

Returns

true if image data are sucessfully read, flase otherwise.

See also

[prepare_inputdata_CPU](#), [prepare_inputdata_GPU](#)

Definition at line 304 of file HologramGenerator.cpp.

7.3 Computing Depth Value

- void [HologramGenerator::GetDepthValues](#) ()
Calculate the physical distances of depth map layers.
- void [HologramGenerator::change_depth_quan_CPU](#) ()
Quantize depth map on the CPU, when the number of depth quantization is not the default value (i.e. `FLAG_CHANGE_DEPTH_QUANTIZATION == 1`).
- void [HologramGenerator::change_depth_quan_GPU](#) ()
Quantize depth map on the GPU, when the number of depth quantization is not the default value (i.e. `FLAG_CHANGE_DEPTH_QUANTIZATION == 1`).

7.3.1 Detailed Description

7.3.2 Function Documentation

7.3.2.1 [change_depth_quan_CPU\(\)](#)

```
void HologramGenerator::change_depth_quan_CPU ( ) [private]
```

Calculate the value of 'depth_index_'.

See also

[GetDepthValues](#)

Definition at line 74 of file `HologramGenerator_CPU.cpp`.

7.3.2.2 [change_depth_quan_GPU\(\)](#)

```
void HologramGenerator::change_depth_quan_GPU ( ) [private]
```

Calculate the value of 'depth_index_gpu_'.

See also

[GetDepthValues](#)

Definition at line 211 of file `HologramGenerator_GPU.cpp`.

7.3.2.3 [GetDepthValues\(\)](#)

```
void HologramGenerator::GetDepthValues ( ) [private]
```

Initialize 'dstep_' & 'dlevel_' variables. If `FLAG_CHANGE_DEPTH_QUANTIZATION == 1`, recalculate 'depth_index_' variable.

See also

[change_depth_quan_CPU](#), [change_depth_quan_GPU](#)

Definition at line 391 of file `HologramGenerator.cpp`.

7.4 Transform

- void [HologramGenerator::TransformViewingWindow](#) ()

Transform target object to reflect the system configuration of holographic display.

7.4.1 Detailed Description

7.4.2 Function Documentation

7.4.2.1 TransformViewingWindow()

```
void HologramGenerator::TransformViewingWindow ( ) [private]
```

Calculate 'dlevel_transform_' variable by using 'field_lens' & 'dlevel_'.

Definition at line 423 of file HologramGenerator.cpp.

7.5 Generation Hologram

Functions

- void [HologramGenerator::GenerateHologram](#) ()
Generate a hologram, main funtion.
- void [HologramGenerator::Calc_Holo_by_Depth](#) (int frame)
Generate a hologram.
- void [HologramGenerator::Calc_Holo_CPU](#) (int frame)
Main method for generating a hologram on the CPU.
- void [HologramGenerator::Calc_Holo_GPU](#) (int frame)
Main method for generating a hologram on the GPU.
- void [HologramGenerator::Propagation_AngularSpectrum_CPU](#) (Complex *input_u, double propagation_dist)
Angular spectrum propagation method for CPU implementation.
- void [HologramGenerator::Propagation_AngularSpectrum_GPU](#) (cufftDoubleComplex *input_u, double propagation_dist)
Angular spectrum propagation method for GPU implementation.

7.5.1 Detailed Description

7.5.2 Function Documentation

7.5.2.1 Calc_Holo_by_Depth()

```
void HologramGenerator::Calc_Holo_by_Depth (
    int frame ) [private]
```

Parameters

<i>frame</i>	: the frame number of the image.
--------------	----------------------------------

See also

[Calc_Holo_CPU](#), [Calc_Holo_GPU](#)

Definition at line 442 of file HologramGenerator.cpp.

7.5.2.2 Calc_Holo_CPU()

```
void HologramGenerator::Calc_Holo_CPU (
    int frame ) [private]
```

For each depth level,

1. find each depth plane of the input image.
2. apply carrier phase delay.
3. propagate it to the hologram plan.
4. accumulate the result of each propagation.

The final result is accumulated in the variable 'U_complex_'.

Parameters

<i>frame</i>	: the frame number of the image.
--------------	----------------------------------

See also

[Calc_Holo_by_Depth](#), [Propagation_AngularSpectrum_CPU](#)

Definition at line 117 of file HologramGenerator_CPU.cpp.

7.5.2.3 Calc_Holo_GPU()

```
void HologramGenerator::Calc_Holo_GPU (
    int frame ) [private]
```

For each depth level,

1. find each depth plane of the input image.
2. apply carrier phase delay.
3. propagate it to the hologram plan.
4. accumulate the result of each propagation.

It uses CUDA kernels, `cudaDepthHoloKernel` & `cudaPropagation_AngularSpKernel`.
The final result is accumulated in the variable 'u_complex_gpu_'.

Parameters

<i>frame</i>	: the frame number of the image.
--------------	----------------------------------

See also

[Calc_Holo_by_Depth](#), [Propagation_AngularSpectrum_GPU](#)

Definition at line 246 of file HologramGenerator_GPU.cpp.

7.5.2.4 GenerateHologram()

```
void HologramGenerator::GenerateHologram ( )
```

For each frame,

1. Read image depth data.
2. Compute the physical distance of depth map.
3. Transform target object to reflect the system configuration of holographic display.
4. Generate a hologram.
5. Encode the generated hologram.
6. Write the hologram to a image.

See also

[ReadImageDepth](#), [GetDepthValues](#), [TransformViewingWindow](#), [Calc_Holo_by_Depth](#), [Encoding_↔
Symmetrization](#), [Write_Result_image](#)

Definition at line 261 of file HologramGenerator.cpp.

7.5.2.5 Propagation_AngularSpectrum_CPU()

```
void HologramGenerator::Propagation_AngularSpectrum_CPU (
    Complex * input_u,
    double propagation_dist ) [private]
```

The propagation results of all depth levels are accumulated in the variable 'U_complex'.

Parameters

<i>input_u</i>	: each depth plane data.
<i>propagation_dist</i>	: the distance from the object to the hologram plane.

See also

[Calc_Holo_by_Depth](#), [Calc_Holo_CPU](#), [fftwShift](#)

Definition at line 186 of file HologramGenerator_CPU.cpp.

7.5.2.6 Propagation_AngularSpectrum_GPU()

```
void HologramGenerator::Propagation_AngularSpectrum_GPU (
    cufftDoubleComplex * input_u,
    double propagation_dist ) [private]
```

The propagation results of all depth levels are accumulated in the variable 'u_complex_gpu_'.

Parameters

<i>input_u</i>	: each depth plane data.
<i>propagation_dist</i>	: the distance from the object to the hologram plane.

See also

[Calc_Holo_by_Depth](#), [Calc_Holo_GPU](#), [cudaFFT](#)

Definition at line 311 of file HologramGenerator_GPU.cpp.

7.6 Encoding

- void [HologramGenerator::Encoding_Symmetrization](#) ([ivec2](#) sig_location)
Encode the CGH according to a signal location parameter.
- void [HologramGenerator::encoding_CPU](#) (int cropx1, int cropx2, int copy1, int copy2, [ivec2](#) sig_location)
Encode the CGH according to a signal location parameter on the CPU.
- void [HologramGenerator::encoding_GPU](#) (int cropx1, int cropx2, int copy1, int copy2, [ivec2](#) sig_location)
Encode the CGH according to a signal location parameter on GPU.

7.6.1 Detailed Description

7.6.2 Function Documentation

7.6.2.1 encoding_CPU()

```
void HologramGenerator::encoding_CPU (
    int cropx1,
    int cropx2,
    int copy1,
    int copy2,
    ivec2 sig_location ) [private]
```

The CPU variable, u255_fringe_ on CPU has the final result.

Parameters

<i>cropx1</i>	: the start x-coordinate to crop.
<i>cropx2</i>	: the end x-coordinate to crop.
<i>copy1</i>	: the start y-coordinate to crop.
<i>copy2</i>	: the end y-coordinate to crop.
<i>sig_location</i>	: ivec2 type, sig_location[0]: upper or lower half, sig_location[1]:left or right half.

See also

[Encoding_Symmetrization](#), [fftwShift](#)

Definition at line 231 of file HologramGenerator_CPU.cpp.

7.6.2.2 encoding_GPU()

```
void HologramGenerator::encoding_GPU (
    int cropx1,
    int cropx2,
```

```

int copy1,
int copy2,
ivec2 sig_location ) [private]

```

The variable, `u255_fringe_` has the final result.

Parameters

<i>cropx1</i>	: the start x-coordinate to crop.
<i>cropx2</i>	: the end x-coordinate to crop.
<i>copy1</i>	: the start y-coordinate to crop.
<i>copy2</i>	: the end y-coordinate to crop.
<i>sig_location</i>	: ivec2 type, sig_location[0]: upper or lower half, sig_location[1]:left or right half.

See also

[Encoding_Symmetrization](#), [cudaCropFringe](#), [cudaFFT](#), [cudaGetFringe](#)

Definition at line 338 of file HologramGenerator_GPU.cpp.

7.6.2.3 Encoding_Symmetrization()

```

void HologramGenerator::Encoding_Symmetrization (
    ivec2 sig_location ) [private]

```

Parameters

<i>sig_location</i>	: ivec2 type, sig_location[0]: upper or lower half, sig_location[1]:left or right half.
---------------------	---

See also

[encoding_CPU](#), [encoding_GPU](#)

Definition at line 481 of file HologramGenerator.cpp.

7.7 Writing Image

- void [HologramGenerator::Write_Result_image](#) (int ftr)
Write the result image.

7.7.1 Detailed Description

7.7.2 Function Documentation

7.7.2.1 Write_Result_image()

```
void HologramGenerator::Write_Result_image (  
    int ftr ) [private]
```

Parameters

<i>ftr</i>	: the frame number of the image.
------------	----------------------------------

Definition at line 528 of file HologramGenerator.cpp.

7.8 Reconstruction

Functions

- void [HologramGenerator::ReconstructImage](#) ()
It is a testing function used for the reconstruction.
- void [HologramGenerator::Reconstruction](#) (fftw_complex *in, fftw_complex *out)
It is a testing function used for the reconstruction.
- void [HologramGenerator::Test_Propagation_to_Eye_Pupil](#) (fftw_complex *in, fftw_complex *out)
It is a testing function used for the reconstruction.
- void [HologramGenerator::Write_Simulation_image](#) (int num, double val)
It is a testing function used for the reconstruction.
- void [HologramGenerator::circshift](#) (Complex *in, Complex *out, int shift_x, int shift_y, int nx, int ny)
It is a testing function used for the reconstruction.

7.8.1 Detailed Description

7.8.2 Function Documentation

7.8.2.1 circshift()

```
void HologramGenerator::circshift (
    Complex * in,
    Complex * out,
    int shift_x,
    int shift_y,
    int nx,
    int ny ) [private]
```

Definition at line 663 of file HologramGenerator_CPU.cpp.

7.8.2.2 ReconstructImage()

```
void HologramGenerator::ReconstructImage ( )
```

Definition at line 415 of file HologramGenerator_CPU.cpp.

7.8.2.3 Reconstruction()

```
void HologramGenerator::Reconstruction (
    fftw_complex * in,
    fftw_complex * out ) [private]
```

Definition at line 544 of file HologramGenerator_CPU.cpp.

7.8.2.4 Test_Propagation_to_Eye_Pupil()

```
void HologramGenerator::Test_Propagation_to_Eye_Pupil (
    fftw_complex * in,
    fftw_complex * out ) [private]
```

Definition at line 494 of file HologramGenerator_CPU.cpp.

7.8.2.5 Write_Simulation_image()

```
void HologramGenerator::Write_Simulation_image (
    int num,
    double val ) [private]
```

Definition at line 619 of file HologramGenerator_CPU.cpp.

7.9 GPU Modules

Functions

- void [cudaFFT](#) (CUstream_st *stream, int nx, int ny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int direction, bool bNormalized=false)
Convert data from the spatial domain to the frequency domain using 2D FFT on GPU.
- void [cudaCropFringe](#) (CUstream_st *stream, int nx, int ny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int cropx1, int cropx2, int cropy1, int cropy2)
Crop input data according to x, y coordinates on GPU.
- void [cudaDepthHoloKernel](#) (CUstream_st *stream, int pnx, int pny, cufftDoubleComplex *u_o_gpu_, unsigned char *img_src_gpu_, unsigned char *dimg_src_gpu_, double *depth_index_gpu_, int dtr, double rand_phase_val_a, double rand_phase_val_b, double carrier_phase_delay_a, double carrier_phase_delay_b, int flag_change_depth_quan, unsigned int default_depth_quan)
Find each depth plane of the input image and apply carrier phase delay to it on GPU.
- void [cudaPropagation_AngularSpKernel](#) (CUstream_st *stream_, int pnx, int pny, cufftDoubleComplex *input_d, cufftDoubleComplex *u_complex, double ppx, double ppy, double ssx, double ssy, double lambda, double params_k, double propagation_dist)
Angular spectrum propagation method for GPU implementation.
- void [cudaGetFringe](#) (CUstream_st *stream, int pnx, int pny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int sig_locationx, int sig_locationy, double ssx, double ssy, double ppx, double ppy, double PI)
Encode the CGH according to a signal location parameter on the GPU.
- void [cudaChangeDepthQuanKernel](#) (CUstream_st *stream_, int pnx, int pny, double *depth_index_gpu, unsigned char *dimg_src_gpu, int dtr, double d1, double d2, double params_num_of_depth, double params_far_depthmap, double params_near_depthmap)
Quantize depth map on the GPU, only when the number of depth quantization is not the default value (i.e. FLAG_CHANGE_DEPTH_QUANTIZATION == 1).

7.9.1 Detailed Description

7.9.2 Function Documentation

7.9.2.1 [cudaChangeDepthQuanKernel\(\)](#)

```
void cudaChangeDepthQuanKernel (
    CUstream_st * stream_,
    int pnx,
    int pny,
    double * depth_index_gpu,
    unsigned char * dimg_src_gpu,
    int dtr,
    double d1,
    double d2,
    double params_num_of_depth,
    double params_far_depthmap,
    double params_near_depthmap )
```

Calculate the value of 'depth_index_gpu'.

Parameters

<i>stream</i>	: CUDA Stream
<i>pnx</i>	: the number of column of the input data
<i>pny</i>	: the number of row of the input data
<i>depth_index_gpu</i>	: output variable
<i>dimg_src_gpu</i>	: input depth map data
<i>dtr</i>	: the current working depth level
<i>d1</i>	: the starting physical point of each depth level
<i>d2</i>	: the ending physical point of each depth level
<i>params_num_of_depth</i>	: the number of depth level
<i>params_far_depthmap</i>	: NEAR_OF_DEPTH_MAP at config file
<i>params_near_depthmap</i>	: FAR_OF_DEPTH_MAP at config file

See also

change_depth_quan_GPU

7.9.2.2 cudaCropFringe()

```
void cudaCropFringe (
    CUstream_st * stream,
    int nx,
    int ny,
    cufftDoubleComplex * in_field,
    cufftDoubleComplex * out_field,
    int cropx1,
    int cropx2,
    int cropy1,
    int cropy2 )
```

call CUDA Kernel - cropFringe.

Parameters

<i>stream</i>	: CUDA Stream
<i>nx</i>	: the number of column of the input data
<i>ny</i>	: the number of row of the input data
<i>in_field</i>	: input complex data variable
<i>output_field</i>	: output complex data variable
<i>cropx1</i>	: the start x-coordinate to crop.
<i>cropx2</i>	: the end x-coordinate to crop.
<i>cropy1</i>	: the start y-coordinate to crop.
<i>cropy2</i>	: the end y-coordinate to crop.

See also

`encoding_GPU`

7.9.2.3 `cudaDepthHoloKernel()`

```
void cudaDepthHoloKernel (
    CUstream_st * stream,
    int pnx,
    int pny,
    cufftDoubleComplex * u_o_gpu_,
    unsigned char * img_src_gpu_,
    unsigned char * dimg_src_gpu_,
    double * depth_index_gpu_,
    int dtr,
    double rand_phase_val_a,
    double rand_phase_val_b,
    double carrier_phase_delay_a,
    double carrier_phase_delay_b,
    int flag_change_depth_quan,
    unsigned int default_depth_quan )
```

call CUDA Kernel - `depth_sources_kernel`.

Parameters

<i>stream</i>	: CUDA Stream
<i>nx</i>	: the number of column of the input data
<i>ny</i>	: the number of row of the input data
<i>u_o_gpu_</i>	: output variable
<i>img_src_gpu_</i>	: input image data
<i>dimg_src_gpu_</i>	: input depth map data
<i>depth_index_gpu_</i>	: input quantized depth map data
<i>dtr</i>	: current working depth level
<i>rand_phase_val_a</i>	: the real part of the random phase value
<i>rand_phase_val_b</i>	: the imaginary part of the random phase value
<i>carrier_phase_delay_a</i>	: the real part of the carrier phase delay
<i>carrier_phase_delay_b</i>	: the imaginary part of the carrier phase delay
<i>flag_change_depth_quan</i>	: if true, change the depth quantization from the default value.
<i>default_depth_quan</i>	: default value of the depth quantization - 256

See also

`Calc_Holo_GPU`

7.9.2.4 cudaFFT()

```
void cudaFFT (
    CUstream_st * stream,
    int nx,
    int ny,
    cufftDoubleComplex * in_field,
    cufftDoubleComplex * output_field,
    int direction,
    bool bNormailized = false )
```

call CUDA Kernel - fftShift and CUFFT Library.

Parameters

<i>stream</i>	: CUDA Stream
<i>nx</i>	: the number of column of the input data
<i>ny</i>	: the number of row of the input data
<i>in_field</i>	: input complex data variable
<i>output_field</i>	: output complex data variable
<i>direction</i>	: If direction == -1, forward FFT, if type == 1, inverse FFT.
<i>bNomarlized</i>	: If bNomarlized == true, normalize the result after FFT.

See also

Propagation_AngularSpectrum_GPU, encoding_GPU

7.9.2.5 cudaGetFringe()

```
void cudaGetFringe (
    CUstream_st * stream,
    int pnx,
    int pny,
    cufftDoubleComplex * in_field,
    cufftDoubleComplex * out_field,
    int sig_locationx,
    int sig_locationy,
    double ssx,
    double ssy,
    double ppx,
    double ppy,
    double PI )
```

The variable, u255_fringe_ has the final result.

Parameters

<i>stream</i>	: CUDA Stream
<i>pnx</i>	: the number of column of the input data
<i>pny</i>	: the number of row of the input data

Parameters

<i>in_field</i>	: input data
<i>out_field</i>	: output data
<i>sig_locationx</i>	: signal location of x-axis, left or right half
<i>sig_locationy</i>	: signal location of y-axis, upper or lower half
<i>ssx</i>	: $pnx * ppx$
<i>ssy</i>	: $pnx * ppy$
<i>ppx</i>	: pixel pitch of x-axis
<i>ppy</i>	: pixel pitch of y-axis
<i>PI</i>	: π

See also

encoding_GPU

7.9.2.6 cudaPropagation_AngularSpKernel()

```
void cudaPropagation_AngularSpKernel (
    CUstream_st * stream_,
    int pnx,
    int pny,
    cufftDoubleComplex * input_d,
    cufftDoubleComplex * u_complex,
    double ppx,
    double ppy,
    double ssx,
    double ssy,
    double lambda,
    double params_k,
    double propagation_dist )
```

The propagation results of all depth levels are accumulated in the variable 'u_complex_gpu'.

Parameters

<i>stream</i>	: CUDA Stream
<i>pnx</i>	: the number of column of the input data
<i>pny</i>	: the number of row of the input data
<i>input_d</i>	: input data
<i>u_complex</i>	: output data
<i>ppx</i>	: pixel pitch of x-axis
<i>ppy</i>	: pixel pitch of y-axis
<i>ssx</i>	: $pnx * ppx$
<i>ssy</i>	: $pny * ppy$
<i>lambda</i>	: wavelength
<i>params_k</i>	: $2 * \pi / \lambda$
<i>propagation_dist</i>	: the distance from the object to the hologram plane

See also

[Propagation_AngularSpectrum_GPU](#)

Chapter 8

Namespace Documentation

8.1 graphics Namespace Reference

Classes

- struct [ivec2](#)
structure for 2-dimensional integer vector and its arithmetic.
- struct [ivec3](#)
structure for 3-dimensional integer vector and its arithmetic.
- struct [ivec4](#)
structure for 4-dimensional integer vector and its arithmetic.
- struct [vec2](#)
structure for 2-dimensional real type vector and its arithmetic.
- struct [vec3](#)
structure for 3-dimensional real type vector and its arithmetic.
- struct [vec4](#)
structure for 4-dimensional real type vector and its arithmetic.

Functions

- void [set_u_epsilon](#) (real a)
- void [reset_u_epsilon](#) ()
- void [set_zero_epsilon](#) (real a)
- void [reset_zero_epsilon](#) ()
- int [apx_equal](#) (real x, real y)
- int [apx_equal](#) (real x, real y, real eps)
- [ivec2 operator+](#) (const [ivec2](#) &a, const [ivec2](#) &b)
- [ivec2 operator+](#) (int a, const [ivec2](#) &b)
- [ivec2 operator+](#) (const [ivec2](#) &a, int b)
- [ivec2 operator-](#) (const [ivec2](#) &a, const [ivec2](#) &b)
- [ivec2 operator-](#) (int a, const [ivec2](#) &b)
- [ivec2 operator-](#) (const [ivec2](#) &a, int b)
- [ivec2 operator*](#) (const [ivec2](#) &a, const [ivec2](#) &b)
- [ivec2 operator*](#) (int a, const [ivec2](#) &b)
- [ivec2 operator*](#) (const [ivec2](#) &a, int b)
- [ivec2 operator+=](#) ([ivec2](#) &a, const [ivec2](#) &b)

- `ivec2 operator+=` (`ivec2` &a, int b)
- `ivec2 operator-=` (`ivec2` &a, const `ivec2` &b)
- `ivec2 operator-=` (`ivec2` &a, int b)
- `ivec2 operator*=` (`ivec2` &a, const `ivec2` &b)
- `ivec2 operator*=` (`ivec2` &a, int b)
- `int operator==` (const `ivec2` &a, const `ivec2` &b)
- `int operator==` (int a, const `ivec2` &b)
- `int operator==` (const `ivec2` &a, int b)
- `int operator<` (const `ivec2` &a, const `ivec2` &b)
- `int operator<` (int a, const `ivec2` &b)
- `int operator<` (const `ivec2` &a, int b)
- `int operator<=` (const `ivec2` &a, const `ivec2` &b)
- `int operator<=` (int a, const `ivec2` &b)
- `int operator<=` (const `ivec2` &a, int b)
- `int operator>` (const `ivec2` &a, const `ivec2` &b)
- `int operator>` (int a, const `ivec2` &b)
- `int operator>` (const `ivec2` &a, int b)
- `int operator>=` (const `ivec2` &a, const `ivec2` &b)
- `int operator>=` (int a, const `ivec2` &b)
- `int operator>=` (const `ivec2` &a, int b)
- `int operator!=` (const `ivec2` &a, const `ivec2` &b)
- `ivec2 operator-` (const `ivec2` &a)
- `ivec3 operator+` (const `ivec3` &a, const `ivec3` &b)
- `ivec3 operator+` (int a, const `ivec3` &b)
- `ivec3 operator+` (const `ivec3` &a, int b)
- `ivec3 operator-` (const `ivec3` &a, const `ivec3` &b)
- `ivec3 operator-` (int a, const `ivec3` &b)
- `ivec3 operator-` (const `ivec3` &a, int b)
- `ivec3 operator*` (const `ivec3` &a, const `ivec3` &b)
- `ivec3 operator*` (int a, const `ivec3` &b)
- `ivec3 operator*` (const `ivec3` &a, int b)
- `ivec3 operator+=` (`ivec3` &a, const `ivec3` &b)
- `ivec3 operator+=` (`ivec3` &a, int b)
- `ivec3 operator-=` (`ivec3` &a, const `ivec3` &b)
- `ivec3 operator-=` (`ivec3` &a, int b)
- `ivec3 operator*=` (`ivec3` &a, const `ivec3` &b)
- `ivec3 operator*=` (`ivec3` &a, int b)
- `int operator==` (const `ivec3` &a, const `ivec3` &b)
- `int operator==` (int a, const `ivec3` &b)
- `int operator==` (const `ivec3` &a, int b)
- `int operator<` (const `ivec3` &a, const `ivec3` &b)
- `int operator<` (int a, const `ivec3` &b)
- `int operator<` (const `ivec3` &a, int b)
- `int operator<=` (const `ivec3` &a, const `ivec3` &b)
- `int operator<=` (int a, const `ivec3` &b)
- `int operator<=` (const `ivec3` &a, int b)
- `int operator>` (const `ivec3` &a, const `ivec3` &b)
- `int operator>` (int a, const `ivec3` &b)
- `int operator>` (const `ivec3` &a, int b)
- `int operator>=` (const `ivec3` &a, const `ivec3` &b)
- `int operator>=` (int a, const `ivec3` &b)
- `int operator>=` (const `ivec3` &a, int b)
- `int operator!=` (const `ivec3` &a, const `ivec3` &b)
- `ivec3 operator-` (const `ivec3` &a)
- `ivec4 operator+` (const `ivec4` &a, const `ivec4` &b)

- `ivec4 operator+` (int a, const `ivec4` &b)
- `ivec4 operator+` (const `ivec4` &a, int b)
- `ivec4 operator-` (const `ivec4` &a, const `ivec4` &b)
- `ivec4 operator-` (int a, const `ivec4` &b)
- `ivec4 operator-` (const `ivec4` &a, int b)
- `ivec4 operator*` (const `ivec4` &a, const `ivec4` &b)
- `ivec4 operator*` (int a, const `ivec4` &b)
- `ivec4 operator*` (const `ivec4` &a, int b)
- `ivec4 operator+=` (`ivec4` &a, const `ivec4` &b)
- `ivec4 operator+=` (`ivec4` &a, int b)
- `ivec4 operator-=` (`ivec4` &a, const `ivec4` &b)
- `ivec4 operator-=` (`ivec4` &a, int b)
- `ivec4 operator*=` (`ivec4` &a, const `ivec4` &b)
- `ivec4 operator*=` (`ivec4` &a, int b)
- `int operator==` (const `ivec4` &a, const `ivec4` &b)
- `int operator==` (int a, const `ivec4` &b)
- `int operator==` (const `ivec4` &a, int b)
- `int operator<` (const `ivec4` &a, const `ivec4` &b)
- `int operator<` (int a, const `ivec4` &b)
- `int operator<` (const `ivec4` &a, int b)
- `int operator<=` (const `ivec4` &a, const `ivec4` &b)
- `int operator<=` (int a, const `ivec4` &b)
- `int operator<=` (const `ivec4` &a, int b)
- `int operator>` (const `ivec4` &a, const `ivec4` &b)
- `int operator>` (int a, const `ivec4` &b)
- `int operator>` (const `ivec4` &a, int b)
- `int operator>=` (const `ivec4` &a, const `ivec4` &b)
- `int operator!=` (const `ivec4` &a, const `ivec4` &b)
- `int operator>=` (int a, const `ivec4` &b)
- `int operator>=` (const `ivec4` &a, int b)
- `ivec4 operator-` (const `ivec4` &a)
- `void store` (FILE *fp, const `vec2` &v)
- `int scan` (FILE *fp, const `vec2` &v)
- `int apx_equal` (const `vec2` &a, const `vec2` &b)
- `int apx_equal` (const `vec2` &a, const `vec2` &b, `real` eps)
- `void store` (FILE *fp, const `vec3` &v)
- `int scan` (FILE *fp, const `vec3` &v)
- `int apx_equal` (const `vec3` &a, const `vec3` &b)
- `int apx_equal` (const `vec3` &a, const `vec3` &b, `real` eps)
- `void store` (FILE *fp, const `vec4` &v)
- `int scan` (FILE *fp, const `vec4` &v)
- `int apx_equal` (const `vec4` &a, const `vec4` &b)
- `int apx_equal` (const `vec4` &a, const `vec4` &b, `real` eps)
- `vec3 cross` (const `vec3` &a, const `vec3` &b)
- `vec2 operator+` (const `vec2` &a, const `vec2` &b)
- `vec2 operator+` (`real` a, const `vec2` &b)
- `vec2 operator+` (const `vec2` &a, `real` b)
- `vec2 operator-` (const `vec2` &a, const `vec2` &b)
- `vec2 operator-` (`real` a, const `vec2` &b)
- `vec2 operator-` (const `vec2` &a, `real` b)
- `vec2 operator*` (const `vec2` &a, const `vec2` &b)
- `vec2 operator*` (`real` a, const `vec2` &b)
- `vec2 operator*` (const `vec2` &a, `real` b)
- `vec2 operator/` (const `vec2` &a, const `vec2` &b)
- `vec2 operator/` (`real` a, const `vec2` &b)

- `vec2 operator/` (const `vec2` &a, `real` b)
- `vec2 operator+=` (`vec2` &a, const `vec2` &b)
- `vec2 operator+=` (`vec2` &a, `real` b)
- `vec2 operator-=` (`vec2` &a, const `vec2` &b)
- `vec2 operator-=` (`vec2` &a, `real` b)
- `vec2 operator*=` (`vec2` &a, const `vec2` &b)
- `vec2 operator*=` (`vec2` &a, `real` b)
- `vec2 operator/=` (`vec2` &a, const `vec2` &b)
- `vec2 operator/=` (`vec2` &a, `real` b)
- `int operator==` (const `vec2` &a, const `vec2` &b)
- `int operator==` (`real` a, const `vec2` &b)
- `int operator==` (const `vec2` &a, `real` b)
- `int operator<` (const `vec2` &a, const `vec2` &b)
- `int operator<` (`real` a, const `vec2` &b)
- `int operator<` (const `vec2` &a, `real` b)
- `int operator<=` (const `vec2` &a, const `vec2` &b)
- `int operator<=` (`real` a, const `vec2` &b)
- `int operator<=` (const `vec2` &a, `real` b)
- `int operator>` (const `vec2` &a, const `vec2` &b)
- `int operator>` (`real` a, const `vec2` &b)
- `int operator>` (const `vec2` &a, `real` b)
- `int operator>=` (const `vec2` &a, const `vec2` &b)
- `int operator>=` (`real` a, const `vec2` &b)
- `int operator>=` (const `vec2` &a, `real` b)
- `vec2 operator-` (const `vec2` &a)
- `real sum` (const `vec2` &a)
- `real inner` (const `vec2` &a, const `vec2` &b)
- `real norm` (const `vec2` &a)
- `real squaredNorm` (const `vec2` &a)
- `vec2 unit` (const `vec2` &a)
- `real angle` (const `vec2` &a, const `vec2` &b)
- `vec2 proj` (const `vec2` &axis, const `vec2` &a)
- `vec2 absolute` (const `vec2` &val)
- `vec3 operator+` (const `vec3` &a, const `vec3` &b)
- `vec3 operator+` (`real` a, const `vec3` &b)
- `vec3 operator+` (const `vec3` &a, `real` b)
- `vec3 operator-` (const `vec3` &a, const `vec3` &b)
- `vec3 operator-` (`real` a, const `vec3` &b)
- `vec3 operator-` (const `vec3` &a, `real` b)
- `vec3 operator*` (const `vec3` &a, const `vec3` &b)
- `vec3 operator*` (`real` a, const `vec3` &b)
- `vec3 operator*` (const `vec3` &a, `real` b)
- `vec3 operator/` (const `vec3` &a, const `vec3` &b)
- `vec3 operator/` (`real` a, const `vec3` &b)
- `vec3 operator/` (const `vec3` &a, `real` b)
- `vec3 operator+=` (`vec3` &a, const `vec3` &b)
- `vec3 operator+=` (`vec3` &a, `real` b)
- `vec3 operator-=` (`vec3` &a, const `vec3` &b)
- `vec3 operator-=` (`vec3` &a, `real` b)
- `vec3 operator*=` (`vec3` &a, const `vec3` &b)
- `vec3 operator*=` (`vec3` &a, `real` b)
- `vec3 operator/=` (`vec3` &a, const `vec3` &b)
- `vec3 operator/=` (`vec3` &a, `real` b)
- `int operator==` (const `vec3` &a, const `vec3` &b)
- `int operator==` (`real` a, const `vec3` &b)

- `int operator== (const vec3 &a, real b)`
- `int operator< (const vec3 &a, const vec3 &b)`
- `int operator< (real a, const vec3 &b)`
- `int operator< (const vec3 &a, real b)`
- `int operator<= (const vec3 &a, const vec3 &b)`
- `int operator<= (real a, const vec3 &b)`
- `int operator<= (const vec3 &a, real b)`
- `int operator> (const vec3 &a, const vec3 &b)`
- `int operator> (real a, const vec3 &b)`
- `int operator> (const vec3 &a, real b)`
- `int operator>= (const vec3 &a, const vec3 &b)`
- `int operator>= (real a, const vec3 &b)`
- `int operator>= (const vec3 &a, real b)`
- `vec3 operator- (const vec3 &a)`
- `vec3 absolute (const vec3 &val)`
- `real sum (const vec3 &a)`
- `real inner (const vec3 &a, const vec3 &b)`
- `real squaredNorm (const vec3 &a)`
- `real norm (const vec3 &a)`
- `vec3 unit (const vec3 &a)`
- `real angle (const vec3 &a, const vec3 &b)`
- `vec3 proj (const vec3 &axis, const vec3 &a)`
- `vec4 operator+ (const vec4 &a, const vec4 &b)`
- `vec4 operator+ (real a, const vec4 &b)`
- `vec4 operator+ (const vec4 &a, real b)`
- `vec4 operator- (const vec4 &a, const vec4 &b)`
- `vec4 operator- (real a, const vec4 &b)`
- `vec4 operator- (const vec4 &a, real b)`
- `vec4 operator* (const vec4 &a, const vec4 &b)`
- `vec4 operator* (real a, const vec4 &b)`
- `vec4 operator* (const vec4 &a, real b)`
- `vec4 operator/ (const vec4 &a, const vec4 &b)`
- `vec4 operator/ (real a, const vec4 &b)`
- `vec4 operator/ (const vec4 &a, real b)`
- `vec4 operator+= (vec4 &a, const vec4 &b)`
- `vec4 operator+= (vec4 &a, real b)`
- `vec4 operator-= (vec4 &a, const vec4 &b)`
- `vec4 operator-= (vec4 &a, real b)`
- `vec4 operator*= (vec4 &a, const vec4 &b)`
- `vec4 operator*= (vec4 &a, real b)`
- `vec4 operator/= (vec4 &a, const vec4 &b)`
- `vec4 operator/= (vec4 &a, real b)`
- `int operator== (const vec4 &a, const vec4 &b)`
- `int operator== (real a, const vec4 &b)`
- `int operator== (const vec4 &a, real b)`
- `int operator< (const vec4 &a, const vec4 &b)`
- `int operator< (real a, const vec4 &b)`
- `int operator< (const vec4 &a, real b)`
- `int operator<= (const vec4 &a, const vec4 &b)`
- `int operator<= (real a, const vec4 &b)`
- `int operator<= (const vec4 &a, real b)`
- `int operator> (const vec4 &a, const vec4 &b)`
- `int operator> (real a, const vec4 &b)`
- `int operator> (const vec4 &a, real b)`
- `int operator>= (const vec4 &a, const vec4 &b)`

- `int operator>= (real a, const vec4 &b)`
- `int operator>= (const vec4 &a, real b)`
- `vec4 operator- (const vec4 &a)`
- `vec4 absolute (const vec4 &val)`
- `real sum (const vec4 &a)`
- `real inner (const vec4 &a, const vec4 &b)`
- `real squaredNorm (const vec4 &a)`
- `real norm (const vec4 &a)`
- `vec4 unit (const vec4 &a)`
- `real angle (const vec4 &a, const vec4 &b)`
- `vec4 proj (const vec4 &axis, const vec4 &a)`

Variables

- `real epsilon = 1.0e-8`
- `real user_epsilon = 1.0e-8`
- `real intersection_epsilon = 1e-6`
- `real sqrt_epsilon = 1.490116119385000000e-8`
- `real unset_value = -1.23432101234321e+308`
- `real zero_tolerance = 1.0e-12`
- `real angle_tolerance = M_PI/180.0`
- `real zero_epsilon = 1.0e-12`
- `real save_zero_epsilon = 1.0e-12`

8.1.1 Function Documentation

8.1.1.1 `absolute()` [1/3]

```
vec2 graphics::absolute (
    const vec2 & val ) [inline]
```

Definition at line 429 of file vec.h.

8.1.1.2 `absolute()` [2/3]

```
vec3 graphics::absolute (
    const vec3 & val ) [inline]
```

Definition at line 793 of file vec.h.

8.1.1.3 absolute() [3/3]

```
vec4 graphics::absolute (  
    const vec4 & val ) [inline]
```

Definition at line 1185 of file vec.h.

8.1.1.4 angle() [1/3]

```
real graphics::angle (  
    const vec2 & a,  
    const vec2 & b ) [inline]
```

Definition at line 412 of file vec.h.

8.1.1.5 angle() [2/3]

```
real graphics::angle (  
    const vec3 & a,  
    const vec3 & b ) [inline]
```

Definition at line 838 of file vec.h.

8.1.1.6 angle() [3/3]

```
real graphics::angle (  
    const vec4 & a,  
    const vec4 & b ) [inline]
```

Definition at line 1229 of file vec.h.

8.1.1.7 apx_equal() [1/8]

```
int graphics::apx_equal (  
    real x,  
    real y )
```

Definition at line 45 of file epsilon.cpp.

8.1.1.8 apx_equal() [2/8]

```
int graphics::apx_equal (
    real x,
    real y,
    real eps )
```

Definition at line 61 of file epsilon.cpp.

8.1.1.9 apx_equal() [3/8]

```
int graphics::apx_equal (
    const vec2 & a,
    const vec2 & b )
```

Definition at line 101 of file vec.cpp.

8.1.1.10 apx_equal() [4/8]

```
int graphics::apx_equal (
    const vec2 & a,
    const vec2 & b,
    real eps )
```

Definition at line 112 of file vec.cpp.

8.1.1.11 apx_equal() [5/8]

```
int graphics::apx_equal (
    const vec3 & a,
    const vec3 & b )
```

Definition at line 318 of file vec.cpp.

8.1.1.12 apx_equal() [6/8]

```
int graphics::apx_equal (
    const vec3 & a,
    const vec3 & b,
    real eps )
```

Definition at line 329 of file vec.cpp.

8.1.1.13 `apx_equal()` [7/8]

```
int graphics::apx_equal (
    const vec4 & a,
    const vec4 & b )
```

Definition at line 377 of file `vec.cpp`.

8.1.1.14 `apx_equal()` [8/8]

```
int graphics::apx_equal (
    const vec4 & a,
    const vec4 & b,
    real eps )
```

Definition at line 388 of file `vec.cpp`.

8.1.1.15 `cross()`

```
vec3 graphics::cross (
    const vec3 & a,
    const vec3 & b )
```

Definition at line 399 of file `vec.cpp`.

8.1.1.16 `inner()` [1/3]

```
real graphics::inner (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 388 of file `vec.h`.

8.1.1.17 `inner()` [2/3]

```
real graphics::inner (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 814 of file `vec.h`.

8.1.1.18 inner() [3/3]

```
real graphics::inner (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1207 of file vec.h.

8.1.1.19 norm() [1/3]

```
real graphics::norm (
    const vec2 & a ) [inline]
```

Definition at line 394 of file vec.h.

8.1.1.20 norm() [2/3]

```
real graphics::norm (
    const vec3 & a ) [inline]
```

Definition at line 824 of file vec.h.

8.1.1.21 norm() [3/3]

```
real graphics::norm (
    const vec4 & a ) [inline]
```

Definition at line 1215 of file vec.h.

8.1.1.22 operator!=() [1/3]

```
int graphics::operator!= (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 313 of file ivec.h.

8.1.1.23 operator!=() [2/3]

```
int graphics::operator!= (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 625 of file ivec.h.

8.1.1.24 operator!=() [3/3]

```
int graphics::operator!= (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 928 of file ivec.h.

8.1.1.25 operator*() [1/18]

```
ivec2 graphics::operator* (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 100 of file ivec.h.

8.1.1.26 operator*() [2/18]

```
ivec2 graphics::operator* (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 107 of file ivec.h.

8.1.1.27 operator*() [3/18]

```
ivec2 graphics::operator* (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 114 of file ivec.h.

8.1.1.28 operator*() [4/18]

```
vec2 graphics::operator* (  
    const vec2 & a,  
    const vec2 & b ) [inline]
```

Definition at line 153 of file vec.h.

8.1.1.29 operator*() [5/18]

```
vec2 graphics::operator* (  
    real a,  
    const vec2 & b ) [inline]
```

Definition at line 160 of file vec.h.

8.1.1.30 operator*() [6/18]

```
vec2 graphics::operator* (  
    const vec2 & a,  
    real b ) [inline]
```

Definition at line 167 of file vec.h.

8.1.1.31 operator*() [7/18]

```
ivec3 graphics::operator* (  
    const ivec3 & a,  
    const ivec3 & b ) [inline]
```

Definition at line 412 of file ivec.h.

8.1.1.32 operator*() [8/18]

```
ivec3 graphics::operator* (  
    int a,  
    const ivec3 & b ) [inline]
```

Definition at line 419 of file ivec.h.

8.1.1.33 operator*() [9/18]

```
ivec3 graphics::operator* (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 426 of file ivec.h.

8.1.1.34 operator*() [10/18]

```
vec3 graphics::operator* (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 572 of file vec.h.

8.1.1.35 operator*() [11/18]

```
vec3 graphics::operator* (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 579 of file vec.h.

8.1.1.36 operator*() [12/18]

```
vec3 graphics::operator* (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 586 of file vec.h.

8.1.1.37 operator*() [13/18]

```
ivec4 graphics::operator* (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 729 of file ivec.h.

8.1.1.38 operator*() [14/18]

```
ivec4 graphics::operator* (
    int a,
    const ivec4 & b ) [inline]
```

Definition at line 736 of file ivec.h.

8.1.1.39 operator*() [15/18]

```
ivec4 graphics::operator* (
    const ivec4 & a,
    int b ) [inline]
```

Definition at line 743 of file ivec.h.

8.1.1.40 operator*() [16/18]

```
vec4 graphics::operator* (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 962 of file vec.h.

8.1.1.41 operator*() [17/18]

```
vec4 graphics::operator* (
    real a,
    const vec4 & b ) [inline]
```

Definition at line 969 of file vec.h.

8.1.1.42 operator*() [18/18]

```
vec4 graphics::operator* (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 976 of file vec.h.

8.1.1.43 operator*=() [1/12]

```
ivec2 graphics::operator*= (
    ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 173 of file ivec.h.

8.1.1.44 operator*=() [2/12]

```
ivec2 graphics::operator*= (
    ivec2 & a,
    int b ) [inline]
```

Definition at line 178 of file ivec.h.

8.1.1.45 operator*=() [3/12]

```
vec2 graphics::operator*= (
    vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 226 of file vec.h.

8.1.1.46 operator*=() [4/12]

```
vec2 graphics::operator*= (
    vec2 & a,
    real b ) [inline]
```

Definition at line 231 of file vec.h.

8.1.1.47 operator*=() [5/12]

```
ivec3 graphics::operator*= (
    ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 485 of file ivec.h.

8.1.1.48 operator*=() [6/12]

```
ivec3 graphics::operator*= (
    ivec3 & a,
    int b ) [inline]
```

Definition at line 490 of file ivec.h.

8.1.1.49 operator*=() [7/12]

```
vec3 graphics::operator*= (
    vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 643 of file vec.h.

8.1.1.50 operator*=() [8/12]

```
vec3 graphics::operator*= (
    vec3 & a,
    real b ) [inline]
```

Definition at line 648 of file vec.h.

8.1.1.51 operator*=() [9/12]

```
ivec4 graphics::operator*= (
    ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 802 of file ivec.h.

8.1.1.52 operator*=() [10/12]

```
ivec4 graphics::operator*= (
    ivec4 & a,
    int b ) [inline]
```

Definition at line 807 of file ivec.h.

8.1.1.53 operator*=() [11/12]

```
vec4 graphics::operator*= (
    vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1035 of file vec.h.

8.1.1.54 operator*=() [12/12]

```
vec4 graphics::operator*= (
    vec4 & a,
    real b ) [inline]
```

Definition at line 1040 of file vec.h.

8.1.1.55 operator+() [1/18]

```
ivec2 graphics::operator+ (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 54 of file ivec.h.

8.1.1.56 operator+() [2/18]

```
ivec2 graphics::operator+ (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 61 of file ivec.h.

8.1.1.57 operator+() [3/18]

```
ivec2 graphics::operator+ (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 68 of file ivec.h.

8.1.1.58 operator+() [4/18]

```
vec2 graphics::operator+ (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 107 of file vec.h.

8.1.1.59 operator+() [5/18]

```
vec2 graphics::operator+ (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 114 of file vec.h.

8.1.1.60 operator+() [6/18]

```
vec2 graphics::operator+ (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 121 of file vec.h.

8.1.1.61 operator+() [7/18]

```
ivec3 graphics::operator+ (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 366 of file ivec.h.

8.1.1.62 operator+() [8/18]

```
ivec3 graphics::operator+ (
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 373 of file ivec.h.

8.1.1.63 operator+() [9/18]

```
ivec3 graphics::operator+ (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 380 of file ivec.h.

8.1.1.64 operator+() [10/18]

```
vec3 graphics::operator+ (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 526 of file vec.h.

8.1.1.65 operator+() [11/18]

```
vec3 graphics::operator+ (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 533 of file vec.h.

8.1.1.66 operator+() [12/18]

```
vec3 graphics::operator+ (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 540 of file vec.h.

8.1.1.67 operator+() [13/18]

```
ivec4 graphics::operator+ (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 683 of file ivec.h.

8.1.1.68 operator+() [14/18]

```
ivec4 graphics::operator+ (  
    int a,  
    const ivec4 & b ) [inline]
```

Definition at line 690 of file ivec.h.

8.1.1.69 operator+() [15/18]

```
ivec4 graphics::operator+ (  
    const ivec4 & a,  
    int b ) [inline]
```

Definition at line 697 of file ivec.h.

8.1.1.70 operator+() [16/18]

```
vec4 graphics::operator+ (  
    const vec4 & a,  
    const vec4 & b ) [inline]
```

Definition at line 916 of file vec.h.

8.1.1.71 operator+() [17/18]

```
vec4 graphics::operator+ (  
    real a,  
    const vec4 & b ) [inline]
```

Definition at line 923 of file vec.h.

8.1.1.72 operator+() [18/18]

```
vec4 graphics::operator+ (  
    const vec4 & a,  
    real b ) [inline]
```

Definition at line 930 of file vec.h.

8.1.1.73 operator+=() [1/12]

```
ivec2 graphics::operator+= (
    ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 149 of file ivec.h.

8.1.1.74 operator+=() [2/12]

```
ivec2 graphics::operator+= (
    ivec2 & a,
    int b ) [inline]
```

Definition at line 154 of file ivec.h.

8.1.1.75 operator+=() [3/12]

```
vec2 graphics::operator+= (
    vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 202 of file vec.h.

8.1.1.76 operator+=() [4/12]

```
vec2 graphics::operator+= (
    vec2 & a,
    real b ) [inline]
```

Definition at line 207 of file vec.h.

8.1.1.77 operator+=() [5/12]

```
ivec3 graphics::operator+= (
    ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 461 of file ivec.h.

8.1.1.78 operator+=() [6/12]

```
ivec3 graphics::operator+= (
    ivec3 & a,
    int b ) [inline]
```

Definition at line 466 of file ivec.h.

8.1.1.79 operator+=() [7/12]

```
vec3 graphics::operator+= (
    vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 619 of file vec.h.

8.1.1.80 operator+=() [8/12]

```
vec3 graphics::operator+= (
    vec3 & a,
    real b ) [inline]
```

Definition at line 624 of file vec.h.

8.1.1.81 operator+=() [9/12]

```
ivec4 graphics::operator+= (
    ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 778 of file ivec.h.

8.1.1.82 operator+=() [10/12]

```
ivec4 graphics::operator+= (
    ivec4 & a,
    int b ) [inline]
```

Definition at line 783 of file ivec.h.

8.1.1.83 operator+=() [11/12]

```
vec4 graphics::operator+= (
    vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1011 of file vec.h.

8.1.1.84 operator+=() [12/12]

```
vec4 graphics::operator+= (
    vec4 & a,
    real b ) [inline]
```

Definition at line 1016 of file vec.h.

8.1.1.85 operator-() [1/24]

```
ivec2 graphics::operator- (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 77 of file ivec.h.

8.1.1.86 operator-() [2/24]

```
ivec2 graphics::operator- (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 84 of file ivec.h.

8.1.1.87 operator-() [3/24]

```
ivec2 graphics::operator- (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 91 of file ivec.h.

8.1.1.88 operator-() [4/24]

```
vec2 graphics::operator- (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 130 of file vec.h.

8.1.1.89 operator-() [5/24]

```
vec2 graphics::operator- (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 137 of file vec.h.

8.1.1.90 operator-() [6/24]

```
vec2 graphics::operator- (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 144 of file vec.h.

8.1.1.91 operator-() [7/24]

```
ivec2 graphics::operator- (
    const ivec2 & a ) [inline]
```

Definition at line 321 of file ivec.h.

8.1.1.92 operator-() [8/24]

```
vec2 graphics::operator- (
    const vec2 & a ) [inline]
```

Definition at line 369 of file vec.h.

8.1.1.93 operator-() [9/24]

```
ivec3 graphics::operator- (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 389 of file ivec.h.

8.1.1.94 operator-() [10/24]

```
ivec3 graphics::operator- (
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 396 of file ivec.h.

8.1.1.95 operator-() [11/24]

```
ivec3 graphics::operator- (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 403 of file ivec.h.

8.1.1.96 operator-() [12/24]

```
vec3 graphics::operator- (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 549 of file vec.h.

8.1.1.97 operator-() [13/24]

```
vec3 graphics::operator- (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 556 of file vec.h.

8.1.1.98 operator-() [14/24]

```
vec3 graphics::operator- (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 563 of file vec.h.

8.1.1.99 operator-() [15/24]

```
ivec3 graphics::operator- (
    const ivec3 & a ) [inline]
```

Definition at line 633 of file ivec.h.

8.1.1.100 operator-() [16/24]

```
ivec4 graphics::operator- (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 706 of file ivec.h.

8.1.1.101 operator-() [17/24]

```
ivec4 graphics::operator- (
    int a,
    const ivec4 & b ) [inline]
```

Definition at line 713 of file ivec.h.

8.1.1.102 operator-() [18/24]

```
ivec4 graphics::operator- (
    const ivec4 & a,
    int b ) [inline]
```

Definition at line 720 of file ivec.h.

8.1.1.103 operator-() [19/24]

```
vec3 graphics::operator- (
    const vec3 & a ) [inline]
```

Definition at line 786 of file vec.h.

8.1.1.104 operator-() [20/24]

```
vec4 graphics::operator- (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 939 of file vec.h.

8.1.1.105 operator-() [21/24]

```
vec4 graphics::operator- (
    real a,
    const vec4 & b ) [inline]
```

Definition at line 946 of file vec.h.

8.1.1.106 operator-() [22/24]

```
ivec4 graphics::operator- (
    const ivec4 & a ) [inline]
```

Definition at line 952 of file ivec.h.

8.1.1.107 operator-() [23/24]

```
vec4 graphics::operator- (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 953 of file vec.h.

8.1.1.108 operator-() [24/24]

```
vec4 graphics::operator- (
    const vec4 & a ) [inline]
```

Definition at line 1178 of file vec.h.

8.1.1.109 operator-=() [1/12]

```
ivec2 graphics::operator-= (
    ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 161 of file ivec.h.

8.1.1.110 operator-=() [2/12]

```
ivec2 graphics::operator-= (
    ivec2 & a,
    int b ) [inline]
```

Definition at line 166 of file ivec.h.

8.1.1.111 operator-=() [3/12]

```
vec2 graphics::operator-= (
    vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 214 of file vec.h.

8.1.1.112 operator-=() [4/12]

```
vec2 graphics::operator-= (
    vec2 & a,
    real b ) [inline]
```

Definition at line 219 of file vec.h.

8.1.1.113 operator-=() [5/12]

```
ivec3 graphics::operator-= (
    ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 473 of file ivec.h.

8.1.1.114 operator-=() [6/12]

```
ivec3 graphics::operator-= (
    ivec3 & a,
    int b ) [inline]
```

Definition at line 478 of file ivec.h.

8.1.1.115 operator-=() [7/12]

```
vec3 graphics::operator-= (
    vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 631 of file vec.h.

8.1.1.116 operator-=() [8/12]

```
vec3 graphics::operator-= (
    vec3 & a,
    real b ) [inline]
```

Definition at line 636 of file vec.h.

8.1.1.117 operator-=() [9/12]

```
ivec4 graphics::operator-= (
    ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 790 of file ivec.h.

8.1.1.118 operator-=() [10/12]

```
ivec4 graphics::operator-= (
    ivec4 & a,
    int b ) [inline]
```

Definition at line 795 of file ivec.h.

8.1.1.119 operator-=() [11/12]

```
vec4 graphics::operator-= (
    vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1023 of file vec.h.

8.1.1.120 operator-=() [12/12]

```
vec4 graphics::operator-= (
    vec4 & a,
    real b ) [inline]
```

Definition at line 1028 of file vec.h.

8.1.1.121 operator/() [1/9]

```
vec2 graphics::operator/ (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 176 of file vec.h.

8.1.1.122 operator/() [2/9]

```
vec2 graphics::operator/ (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 183 of file vec.h.

8.1.1.123 operator/() [3/9]

```
vec2 graphics::operator/ (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 190 of file vec.h.

8.1.1.124 operator/() [4/9]

```
vec3 graphics::operator/ (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 595 of file vec.h.

8.1.1.125 operator/() [5/9]

```
vec3 graphics::operator/ (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 602 of file vec.h.

8.1.1.126 operator/() [6/9]

```
vec3 graphics::operator/ (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 609 of file vec.h.

8.1.1.127 operator/() [7/9]

```
vec4 graphics::operator/ (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 985 of file vec.h.

8.1.1.128 operator/() [8/9]

```
vec4 graphics::operator/ (
    real a,
    const vec4 & b ) [inline]
```

Definition at line 992 of file vec.h.

8.1.1.129 operator/() [9/9]

```
vec4 graphics::operator/ (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 999 of file vec.h.

8.1.1.130 operator/=() [1/6]

```
vec2 graphics::operator/= (
    vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 238 of file vec.h.

8.1.1.131 operator/=() [2/6]

```
vec2 graphics::operator/= (
    vec2 & a,
    real b ) [inline]
```

Definition at line 243 of file vec.h.

8.1.1.132 operator/=() [3/6]

```
vec3 graphics::operator/= (
    vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 655 of file vec.h.

8.1.1.133 operator/=() [4/6]

```
vec3 graphics::operator/= (
    vec3 & a,
    real b ) [inline]
```

Definition at line 660 of file vec.h.

8.1.1.134 operator/=() [5/6]

```
vec4 graphics::operator/= (
    vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1047 of file vec.h.

8.1.1.135 operator/=() [6/6]

```
vec4 graphics::operator/= (
    vec4 & a,
    real b ) [inline]
```

Definition at line 1052 of file vec.h.

8.1.1.136 operator<() [1/18]

```
int graphics::operator< (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 223 of file ivec.h.

8.1.1.137 operator<() [2/18]

```
int graphics::operator< (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 230 of file ivec.h.

8.1.1.138 operator<>() [3/18]

```
int graphics::operator< (  
    const ivec2 & a,  
    int b ) [inline]
```

Definition at line 237 of file ivec.h.

8.1.1.139 operator<>() [4/18]

```
int graphics::operator< (  
    const vec2 & a,  
    const vec2 & b ) [inline]
```

Definition at line 276 of file vec.h.

8.1.1.140 operator<>() [5/18]

```
int graphics::operator< (  
    real a,  
    const vec2 & b ) [inline]
```

Definition at line 283 of file vec.h.

8.1.1.141 operator<>() [6/18]

```
int graphics::operator< (  
    const vec2 & a,  
    real b ) [inline]
```

Definition at line 290 of file vec.h.

8.1.1.142 operator<>() [7/18]

```
int graphics::operator< (  
    const ivec3 & a,  
    const ivec3 & b ) [inline]
```

Definition at line 535 of file ivec.h.

8.1.1.143 `operator<()` [8/18]

```
int graphics::operator< (  
    int a,  
    const ivec3 & b ) [inline]
```

Definition at line 542 of file ivec.h.

8.1.1.144 `operator<()` [9/18]

```
int graphics::operator< (  
    const ivec3 & a,  
    int b ) [inline]
```

Definition at line 549 of file ivec.h.

8.1.1.145 `operator<()` [10/18]

```
int graphics::operator< (  
    const vec3 & a,  
    const vec3 & b ) [inline]
```

Definition at line 693 of file vec.h.

8.1.1.146 `operator<()` [11/18]

```
int graphics::operator< (  
    real a,  
    const vec3 & b ) [inline]
```

Definition at line 700 of file vec.h.

8.1.1.147 `operator<()` [12/18]

```
int graphics::operator< (  
    const vec3 & a,  
    real b ) [inline]
```

Definition at line 707 of file vec.h.

8.1.1.148 operator<>() [13/18]

```
int graphics::operator< (  
    const ivec4 & a,  
    const ivec4 & b ) [inline]
```

Definition at line 852 of file ivec.h.

8.1.1.149 operator<>() [14/18]

```
int graphics::operator< (  
    int a,  
    const ivec4 & b ) [inline]
```

Definition at line 859 of file ivec.h.

8.1.1.150 operator<>() [15/18]

```
int graphics::operator< (  
    const ivec4 & a,  
    int b ) [inline]
```

Definition at line 866 of file ivec.h.

8.1.1.151 operator<>() [16/18]

```
int graphics::operator< (  
    const vec4 & a,  
    const vec4 & b ) [inline]
```

Definition at line 1085 of file vec.h.

8.1.1.152 operator<>() [17/18]

```
int graphics::operator< (  
    real a,  
    const vec4 & b ) [inline]
```

Definition at line 1092 of file vec.h.

8.1.1.153 operator<() [18/18]

```
int graphics::operator< (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 1099 of file vec.h.

8.1.1.154 operator<=() [1/18]

```
int graphics::operator<= (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 246 of file ivec.h.

8.1.1.155 operator<=() [2/18]

```
int graphics::operator<= (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 253 of file ivec.h.

8.1.1.156 operator<=() [3/18]

```
int graphics::operator<= (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 260 of file ivec.h.

8.1.1.157 operator<=() [4/18]

```
int graphics::operator<= (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 299 of file vec.h.

8.1.1.158 operator<=() [5/18]

```
int graphics::operator<= (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 306 of file vec.h.

8.1.1.159 operator<=() [6/18]

```
int graphics::operator<= (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 313 of file vec.h.

8.1.1.160 operator<=() [7/18]

```
int graphics::operator<= (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 558 of file ivec.h.

8.1.1.161 operator<=() [8/18]

```
int graphics::operator<= (
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 565 of file ivec.h.

8.1.1.162 operator<=() [9/18]

```
int graphics::operator<= (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 572 of file ivec.h.

8.1.1.163 operator<=() [10/18]

```
int graphics::operator<= (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 716 of file vec.h.

8.1.1.164 operator<=() [11/18]

```
int graphics::operator<= (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 723 of file vec.h.

8.1.1.165 operator<=() [12/18]

```
int graphics::operator<= (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 730 of file vec.h.

8.1.1.166 operator<=() [13/18]

```
int graphics::operator<= (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 875 of file ivec.h.

8.1.1.167 operator<=() [14/18]

```
int graphics::operator<= (
    int a,
    const ivec4 & b ) [inline]
```

Definition at line 882 of file ivec.h.

8.1.1.168 operator<=() [15/18]

```
int graphics::operator<= (
    const ivec4 & a,
    int b ) [inline]
```

Definition at line 889 of file ivec.h.

8.1.1.169 operator<=() [16/18]

```
int graphics::operator<= (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1108 of file vec.h.

8.1.1.170 operator<=() [17/18]

```
int graphics::operator<= (
    real a,
    const vec4 & b ) [inline]
```

Definition at line 1115 of file vec.h.

8.1.1.171 operator<=() [18/18]

```
int graphics::operator<= (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 1122 of file vec.h.

8.1.1.172 operator==([1/18]

```
int graphics::operator==(
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 200 of file ivec.h.

8.1.1.173 operator==() [2/18]

```
int graphics::operator==(
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 207 of file ivec.h.

8.1.1.174 operator==() [3/18]

```
int graphics::operator==(
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 214 of file ivec.h.

8.1.1.175 operator==() [4/18]

```
int graphics::operator==(
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 253 of file vec.h.

8.1.1.176 operator==() [5/18]

```
int graphics::operator==(
    real a,
    const vec2 & b ) [inline]
```

Definition at line 260 of file vec.h.

8.1.1.177 operator==() [6/18]

```
int graphics::operator==(
    const vec2 & a,
    real b ) [inline]
```

Definition at line 267 of file vec.h.

8.1.1.178 operator==([7/18]

```
int graphics::operator==(
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 512 of file ivec.h.

8.1.1.179 operator==([8/18]

```
int graphics::operator==(
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 519 of file ivec.h.

8.1.1.180 operator==([9/18]

```
int graphics::operator==(
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 526 of file ivec.h.

8.1.1.181 operator==([10/18]

```
int graphics::operator==(
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 670 of file vec.h.

8.1.1.182 operator==([11/18]

```
int graphics::operator==(
    real a,
    const vec3 & b ) [inline]
```

Definition at line 677 of file vec.h.

8.1.1.183 operator==() [12/18]

```
int graphics::operator==(
    const vec3 & a,
    real b ) [inline]
```

Definition at line 684 of file vec.h.

8.1.1.184 operator==() [13/18]

```
int graphics::operator==(
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 829 of file ivec.h.

8.1.1.185 operator==() [14/18]

```
int graphics::operator==(
    int a,
    const ivec4 & b ) [inline]
```

Definition at line 836 of file ivec.h.

8.1.1.186 operator==() [15/18]

```
int graphics::operator==(
    const ivec4 & a,
    int b ) [inline]
```

Definition at line 843 of file ivec.h.

8.1.1.187 operator==() [16/18]

```
int graphics::operator==(
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1062 of file vec.h.

8.1.1.188 operator==([17/18]

```
int graphics::operator==(
    real a,
    const vec4 & b ) [inline]
```

Definition at line 1069 of file vec.h.

8.1.1.189 operator==([18/18]

```
int graphics::operator==(
    const vec4 & a,
    real b ) [inline]
```

Definition at line 1076 of file vec.h.

8.1.1.190 operator>() [1/18]

```
int graphics::operator> (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 269 of file ivec.h.

8.1.1.191 operator>() [2/18]

```
int graphics::operator> (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 276 of file ivec.h.

8.1.1.192 operator>() [3/18]

```
int graphics::operator> (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 283 of file ivec.h.

8.1.1.193 operator>() [4/18]

```
int graphics::operator> (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 322 of file vec.h.

8.1.1.194 operator>() [5/18]

```
int graphics::operator> (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 329 of file vec.h.

8.1.1.195 operator>() [6/18]

```
int graphics::operator> (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 336 of file vec.h.

8.1.1.196 operator>() [7/18]

```
int graphics::operator> (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 581 of file ivec.h.

8.1.1.197 operator>() [8/18]

```
int graphics::operator> (
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 588 of file ivec.h.

8.1.1.198 `operator>()` [9/18]

```
int graphics::operator> (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 595 of file ivec.h.

8.1.1.199 `operator>()` [10/18]

```
int graphics::operator> (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 739 of file vec.h.

8.1.1.200 `operator>()` [11/18]

```
int graphics::operator> (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 746 of file vec.h.

8.1.1.201 `operator>()` [12/18]

```
int graphics::operator> (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 753 of file vec.h.

8.1.1.202 `operator>()` [13/18]

```
int graphics::operator> (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 898 of file ivec.h.

8.1.1.203 operator>() [14/18]

```
int graphics::operator> (  
    int a,  
    const ivec4 & b ) [inline]
```

Definition at line 905 of file ivec.h.

8.1.1.204 operator>() [15/18]

```
int graphics::operator> (  
    const ivec4 & a,  
    int b ) [inline]
```

Definition at line 912 of file ivec.h.

8.1.1.205 operator>() [16/18]

```
int graphics::operator> (  
    const vec4 & a,  
    const vec4 & b ) [inline]
```

Definition at line 1131 of file vec.h.

8.1.1.206 operator>() [17/18]

```
int graphics::operator> (  
    real a,  
    const vec4 & b ) [inline]
```

Definition at line 1138 of file vec.h.

8.1.1.207 operator>() [18/18]

```
int graphics::operator> (  
    const vec4 & a,  
    real b ) [inline]
```

Definition at line 1145 of file vec.h.

8.1.1.208 operator>=() [1/18]

```
int graphics::operator>= (
    const ivec2 & a,
    const ivec2 & b ) [inline]
```

Definition at line 293 of file ivec.h.

8.1.1.209 operator>=() [2/18]

```
int graphics::operator>= (
    int a,
    const ivec2 & b ) [inline]
```

Definition at line 299 of file ivec.h.

8.1.1.210 operator>=() [3/18]

```
int graphics::operator>= (
    const ivec2 & a,
    int b ) [inline]
```

Definition at line 306 of file ivec.h.

8.1.1.211 operator>=() [4/18]

```
int graphics::operator>= (
    const vec2 & a,
    const vec2 & b ) [inline]
```

Definition at line 345 of file vec.h.

8.1.1.212 operator>=() [5/18]

```
int graphics::operator>= (
    real a,
    const vec2 & b ) [inline]
```

Definition at line 352 of file vec.h.

8.1.1.213 operator>=() [6/18]

```
int graphics::operator>= (
    const vec2 & a,
    real b ) [inline]
```

Definition at line 359 of file vec.h.

8.1.1.214 operator>=() [7/18]

```
int graphics::operator>= (
    const ivec3 & a,
    const ivec3 & b ) [inline]
```

Definition at line 604 of file ivec.h.

8.1.1.215 operator>=() [8/18]

```
int graphics::operator>= (
    int a,
    const ivec3 & b ) [inline]
```

Definition at line 611 of file ivec.h.

8.1.1.216 operator>=() [9/18]

```
int graphics::operator>= (
    const ivec3 & a,
    int b ) [inline]
```

Definition at line 618 of file ivec.h.

8.1.1.217 operator>=() [10/18]

```
int graphics::operator>= (
    const vec3 & a,
    const vec3 & b ) [inline]
```

Definition at line 762 of file vec.h.

8.1.1.218 operator>=() [11/18]

```
int graphics::operator>= (
    real a,
    const vec3 & b ) [inline]
```

Definition at line 769 of file vec.h.

8.1.1.219 operator>=() [12/18]

```
int graphics::operator>= (
    const vec3 & a,
    real b ) [inline]
```

Definition at line 776 of file vec.h.

8.1.1.220 operator>=() [13/18]

```
int graphics::operator>= (
    const ivec4 & a,
    const ivec4 & b ) [inline]
```

Definition at line 921 of file ivec.h.

8.1.1.221 operator>=() [14/18]

```
int graphics::operator>= (
    int a,
    const ivec4 & b ) [inline]
```

Definition at line 935 of file ivec.h.

8.1.1.222 operator>=() [15/18]

```
int graphics::operator>= (
    const ivec4 & a,
    int b ) [inline]
```

Definition at line 942 of file ivec.h.

8.1.1.223 operator>=() [16/18]

```
int graphics::operator>= (
    const vec4 & a,
    const vec4 & b ) [inline]
```

Definition at line 1154 of file vec.h.

8.1.1.224 operator>=() [17/18]

```
int graphics::operator>= (
    real a,
    const vec4 & b ) [inline]
```

Definition at line 1161 of file vec.h.

8.1.1.225 operator>=() [18/18]

```
int graphics::operator>= (
    const vec4 & a,
    real b ) [inline]
```

Definition at line 1168 of file vec.h.

8.1.1.226 proj() [1/3]

```
vec2 graphics::proj (
    const vec2 & axis,
    const vec2 & a ) [inline]
```

Definition at line 423 of file vec.h.

8.1.1.227 proj() [2/3]

```
vec3 graphics::proj (
    const vec3 & axis,
    const vec3 & a ) [inline]
```

Definition at line 849 of file vec.h.

8.1.1.228 `proj()` [3/3]

```
vec4 graphics::proj (
    const vec4 & axis,
    const vec4 & a ) [inline]
```

Definition at line 1240 of file vec.h.

8.1.1.229 `reset_u_epsilon()`

```
void graphics::reset_u_epsilon ( )
```

Definition at line 27 of file epsilon.cpp.

8.1.1.230 `reset_zero_epsilon()`

```
void graphics::reset_zero_epsilon ( )
```

Definition at line 37 of file epsilon.cpp.

8.1.1.231 `scan()` [1/3]

```
int graphics::scan (
    FILE * fp,
    const vec2 & v )
```

Definition at line 91 of file vec.cpp.

8.1.1.232 `scan()` [2/3]

```
int graphics::scan (
    FILE * fp,
    const vec3 & v )
```

Definition at line 308 of file vec.cpp.

8.1.1.233 scan() [3/3]

```
int graphics::scan (
    FILE * fp,
    const vec4 & v )
```

Definition at line 367 of file vec.cpp.

8.1.1.234 set_u_epsilon()

```
void graphics::set_u_epsilon (
    real a )
```

Definition at line 22 of file epsilon.cpp.

8.1.1.235 set_zero_epsilon()

```
void graphics::set_zero_epsilon (
    real a )
```

Definition at line 31 of file epsilon.cpp.

8.1.1.236 squaredNorm() [1/3]

```
real graphics::squaredNorm (
    const vec2 & a ) [inline]
```

Definition at line 399 of file vec.h.

8.1.1.237 squaredNorm() [2/3]

```
real graphics::squaredNorm (
    const vec3 & a ) [inline]
```

Definition at line 820 of file vec.h.

8.1.1.238 squaredNorm() [3/3]

```
real graphics::squaredNorm (  
    const vec4 & a ) [inline]
```

Definition at line 1212 of file vec.h.

8.1.1.239 store() [1/3]

```
void graphics::store (  
    FILE * fp,  
    const vec2 & v )
```

Definition at line 82 of file vec.cpp.

8.1.1.240 store() [2/3]

```
void graphics::store (  
    FILE * fp,  
    const vec3 & v )
```

Definition at line 299 of file vec.cpp.

8.1.1.241 store() [3/3]

```
void graphics::store (  
    FILE * fp,  
    const vec4 & v )
```

Definition at line 358 of file vec.cpp.

8.1.1.242 sum() [1/3]

```
real graphics::sum (  
    const vec2 & a ) [inline]
```

Definition at line 377 of file vec.h.

8.1.1.243 `sum()` [2/3]

```
real graphics::sum (
    const vec3 & a ) [inline]
```

Definition at line 801 of file vec.h.

8.1.1.244 `sum()` [3/3]

```
real graphics::sum (
    const vec4 & a ) [inline]
```

Definition at line 1192 of file vec.h.

8.1.1.245 `unit()` [1/3]

```
vec2 graphics::unit (
    const vec2 & a ) [inline]
```

Definition at line 403 of file vec.h.

8.1.1.246 `unit()` [2/3]

```
vec3 graphics::unit (
    const vec3 & a ) [inline]
```

Definition at line 829 of file vec.h.

8.1.1.247 `unit()` [3/3]

```
vec4 graphics::unit (
    const vec4 & a ) [inline]
```

Definition at line 1220 of file vec.h.

8.1.2 Variable Documentation

8.1.2.1 angle_tolerance

```
real graphics::angle_tolerance = M_PI/180.0
```

Definition at line 14 of file epsilon.cpp.

8.1.2.2 epsilon

```
real graphics::epsilon = 1.0e-8
```

Definition at line 7 of file epsilon.cpp.

8.1.2.3 intersection_epsilon

```
real graphics::intersection_epsilon = 1e-6
```

Definition at line 9 of file epsilon.cpp.

8.1.2.4 save_zero_epsilon

```
real graphics::save_zero_epsilon = 1.0e-12
```

Definition at line 15 of file epsilon.cpp.

8.1.2.5 sqrt_epsilon

```
real graphics::sqrt_epsilon = 1.490116119385000000e-8
```

Definition at line 10 of file epsilon.cpp.

8.1.2.6 unset_value

```
real graphics::unset_value = -1.23432101234321e+308
```

Definition at line 11 of file epsilon.cpp.

8.1.2.7 user_epsilon

```
real graphics::user_epsilon = 1.0e-8
```

Definition at line 8 of file epsilon.cpp.

8.1.2.8 zero_epsilon

```
real graphics::zero_epsilon = 1.0e-12
```

Definition at line 13 of file epsilon.cpp.

8.1.2.9 zero_tolerance

```
real graphics::zero_tolerance = 1.0e-12
```

Definition at line 12 of file epsilon.cpp.

Chapter 9

Class Documentation

9.1 Complex Class Reference

class for the complex number and its arithmetic.

```
#include <complex.h>
```

Public Member Functions

- `Complex ()`
- `Complex (double ta, double tb)`
- `Complex (const Complex &p)`
- `double mag2 () const`
- `double mag () const`
- `double arg () const`
- `void euler (double &r, double &theta)`
- `Complex conj () const`
- `const Complex & operator= (const Complex &p)`
- `const Complex & operator+= (const Complex &p)`
- `const Complex & operator-= (const Complex &p)`
- `const Complex & operator*= (const double k)`
- `const Complex & operator*= (const Complex &p)`
- `const Complex & operator/= (const double k)`

Public Attributes

- `double a`
- `double b`

Friends

- `const Complex operator+ (const Complex &p, const Complex &q)`
- `const Complex operator- (const Complex &p, const Complex &q)`
- `const Complex operator* (const double k, const Complex &p)`
- `const Complex operator* (const Complex &p, const double k)`
- `const Complex operator* (const Complex &p, const Complex &q)`
- `const Complex operator/ (const Complex &p, const Complex &q)`
- `std::ostream & operator<< (std::ostream &os, const Complex &p)`

9.1.1 Detailed Description

Definition at line 22 of file complex.h.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 `Complex()` [1/3]

```
Complex::Complex ( ) [inline]
```

Definition at line 25 of file complex.h.

9.1.2.2 `Complex()` [2/3]

```
Complex::Complex (
    double ta,
    double tb ) [inline]
```

Definition at line 26 of file complex.h.

9.1.2.3 `Complex()` [3/3]

```
Complex::Complex (
    const Complex & p ) [inline]
```

Definition at line 27 of file complex.h.

9.1.3 Member Function Documentation

9.1.3.1 `arg()`

```
double Complex::arg ( ) const [inline]
```

Definition at line 36 of file complex.h.

9.1.3.2 conj()

```
Complex Complex::conj ( ) const [inline]
```

Definition at line 53 of file complex.h.

9.1.3.3 euler()

```
void Complex::euler (
    double & r,
    double & theta ) [inline]
```

Definition at line 47 of file complex.h.

9.1.3.4 mag()

```
double Complex::mag ( ) const [inline]
```

Definition at line 34 of file complex.h.

9.1.3.5 mag2()

```
double Complex::mag2 ( ) const [inline]
```

Definition at line 33 of file complex.h.

9.1.3.6 operator*=() [1/2]

```
const Complex& Complex::operator*= (
    const double k ) [inline]
```

Definition at line 80 of file complex.h.

9.1.3.7 operator*=() [2/2]

```
const Complex& Complex::operator*= (
    const Complex & p ) [inline]
```

Definition at line 88 of file complex.h.

9.1.3.8 operator+=()

```
const Complex& Complex::operator+= (
    const Complex & p ) [inline]
```

Definition at line 64 of file complex.h.

9.1.3.9 operator-=()

```
const Complex& Complex::operator-= (
    const Complex & p ) [inline]
```

Definition at line 72 of file complex.h.

9.1.3.10 operator/=()

```
const Complex& Complex::operator/= (
    const double k ) [inline]
```

Definition at line 99 of file complex.h.

9.1.3.11 operator=()

```
const Complex& Complex::operator= (
    const Complex & p ) [inline]
```

Definition at line 56 of file complex.h.

9.1.4 Friends And Related Function Documentation

9.1.4.1 operator* [1/3]

```
const Complex operator* (
    const double k,
    const Complex & p ) [friend]
```

Definition at line 117 of file complex.h.

9.1.4.2 `operator*` [2/3]

```
const Complex operator* (
    const Complex & p,
    const double k ) [friend]
```

Definition at line 122 of file complex.h.

9.1.4.3 `operator*` [3/3]

```
const Complex operator* (
    const Complex & p,
    const Complex & q ) [friend]
```

Definition at line 127 of file complex.h.

9.1.4.4 `operator+`

```
const Complex operator+ (
    const Complex & p,
    const Complex & q ) [friend]
```

Definition at line 107 of file complex.h.

9.1.4.5 `operator-`

```
const Complex operator- (
    const Complex & p,
    const Complex & q ) [friend]
```

Definition at line 112 of file complex.h.

9.1.4.6 `operator/`

```
const Complex operator/ (
    const Complex & p,
    const Complex & q ) [friend]
```

Definition at line 132 of file complex.h.

9.1.4.7 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const Complex & p ) [friend]
```

Definition at line 138 of file complex.h.

9.1.5 Member Data Documentation

9.1.5.1 a

```
double Complex::a
```

Definition at line 145 of file complex.h.

9.1.5.2 b

```
double Complex::b
```

Definition at line 145 of file complex.h.

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

- [graphics/complex.h](#)

9.2 HologramDepthmap Class Reference

Test class for executing the sample program, which shows how to use a hologram library.

```
#include <hologramdepthmap.h>
```

Inherits QMainWindow.

Public Member Functions

- [HologramDepthmap](#) (QWidget *parent=0)
- [~HologramDepthmap](#) ()

Private Slots

- void [GenHologram](#) ()
- void [ReconImage](#) ()

Private Attributes

- Ui::HologramDepthmapClass [ui](#)
- [HologramGenerator](#) * [hologram_](#)

9.2.1 Detailed Description

The sample program has a main window form and the user can choose the execution type - CPU and GPU.

Definition at line 13 of file `hologramdepthmap.h`.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 HologramDepthmap()

```
HologramDepthmap::HologramDepthmap (  
    QWidget * parent = 0 )
```

Definition at line 4 of file `hologramdepthmap.cpp`.

9.2.2.2 ~HologramDepthmap()

```
HologramDepthmap::~~HologramDepthmap ( )
```

Definition at line 17 of file `hologramdepthmap.cpp`.

9.2.3 Member Function Documentation

9.2.3.1 GenHologram

```
void HologramDepthmap::GenHologram ( ) [private], [slot]
```

Definition at line 22 of file `hologramdepthmap.cpp`.

9.2.3.2 ReconImage

```
void HologramDepthmap::ReconImage ( ) [private], [slot]
```

Definition at line 47 of file `hologramdepthmap.cpp`.

9.2.4 Member Data Documentation

9.2.4.1 hologram_

`HologramGenerator*` HologramDepthmap::hologram_ [private]

Definition at line 30 of file hologramdepthmap.h.

9.2.4.2 ui

`Ui::HologramDepthmapClass` HologramDepthmap::ui [private]

Definition at line 28 of file hologramdepthmap.h.

The documentation for this class was generated from the following files:

- HologramDepthmap/[hologramdepthmap.h](#)
- HologramDepthmap/[hologramdepthmap.cpp](#)

9.3 HologramGenerator Class Reference

Main class for generating a hologram using depth map data.

```
#include <HologramGenerator.h>
```

Public Member Functions

- [HologramGenerator](#) ()
Constructor.
- [~HologramGenerator](#) ()
Destructor.
- void [setMode](#) (bool isCPU)
Set the value of a variable isCPU_ (true or false)
- bool [readConfig](#) ()
Read parameters from a config file(config_openholo.txt).
- void [initialize](#) ()
Initialize variables for CPU and GPU implementation.
- void [GenerateHologram](#) ()
Generate a hologram, main funtion.
- void [ReconstructImage](#) ()
It is a testing function used for the reconstruction.

Private Member Functions

- void [get_rand_phase_value](#) ([Complex](#) &rand_phase_val)
Assign random phase value if RANDOM_PHASE == 1.
 - void [get_shift_phase_value](#) ([Complex](#) &shift_phase_val, int idx, [ivec2](#) sig_location)
Calculate the shift phase value.
 - void [fftwShift](#) ([Complex](#) *src, [Complex](#) *dst, [fftw_complex](#) *in, [fftw_complex](#) *out, int nx, int ny, int type, bool bNormalized=false)
Convert data from the spatial domain to the frequency domain using 2D FFT on CPU.
 - void [exponent_complex](#) ([Complex](#) *val)
Calculate the exponential of the complex number.
 - void [fftwShift](#) (int nx, int ny, [Complex](#) *input, [Complex](#) *output)
Swap the top-left quadrant of data with the bottom-right , and the top-right quadrant with the bottom-left.
-
- void [init_CPU](#) ()
Initialize variables for the CPU implementation.
 - void [init_GPU](#) ()
Initialize variables for the GPU implementation.
-
- bool [ReadImageDepth](#) (int ftr)
Read image and depth map.
 - bool [prepare_inputdata_CPU](#) (uchar *img, uchar *dimg)
Preprocess input image & depth map data for the CPU implementation.
 - bool [prepare_inputdata_GPU](#) (uchar *img, uchar *dimg)
Copy input image & depth map data into a GPU.
-
- void [GetDepthValues](#) ()
Calculate the physical distances of depth map layers.
 - void [change_depth_quan_CPU](#) ()
Quantize depth map on the CPU, when the number of depth quantization is not the default value (i.e. FLAG_CHANGE_DEPTH_QUANTIZATION == 1).
 - void [change_depth_quan_GPU](#) ()
Quantize depth map on the GPU, when the number of depth quantization is not the default value (i.e. FLAG_CHANGE_DEPTH_QUANTIZATION == 1).
-
- void [TransformViewingWindow](#) ()
Transform target object to reflect the system configuration of holographic display.

- void [Calc_Holo_by_Depth](#) (int frame)
Generate a hologram.
 - void [Calc_Holo_CPU](#) (int frame)
Main method for generating a hologram on the CPU.
 - void [Calc_Holo_GPU](#) (int frame)
Main method for generating a hologram on the GPU.
 - void [Propagation_AngularSpectrum_CPU](#) (Complex *input_u, double propagation_dist)
Angular spectrum propagation method for CPU implementation.
 - void [Propagation_AngularSpectrum_GPU](#) (cufftDoubleComplex *input_u, double propagation_dist)
Angular spectrum propagation method for GPU implementation.
-
- void [Encoding_Symmetrization](#) (ivec2 sig_location)
Encode the CGH according to a signal location parameter.
 - void [encoding_CPU](#) (int cropx1, int cropx2, int copy1, int copy2, ivec2 sig_location)
Encode the CGH according to a signal location parameter on the CPU.
 - void [encoding_GPU](#) (int cropx1, int cropx2, int copy1, int copy2, ivec2 sig_location)
Encode the CGH according to a signal location parameter on GPU.
-
- void [Write_Result_image](#) (int ftr)
Write the result image.
-
- void [Reconstruction](#) (fftw_complex *in, fftw_complex *out)
It is a testing function used for the reconstruction.
 - void [Test_Propagation_to_Eye_Pupil](#) (fftw_complex *in, fftw_complex *out)
It is a testing function used for the reconstruction.
 - void [Write_Simulation_image](#) (int num, double val)
It is a testing function used for the reconstruction.
 - void [circshift](#) (Complex *in, Complex *out, int shift_x, int shift_y, int nx, int ny)
It is a testing function used for the reconstruction.

Private Attributes

- bool [isCPU_](#)
if true, it is implemented on the CPU, otherwise on the GPU.
- unsigned char * [img_src_gpu_](#)
GPU variable - image source data, values are from 0 to 255.
- unsigned char * [dimg_src_gpu_](#)
GPU variable - depth map data, values are from 0 to 255.
- double * [depth_index_gpu_](#)
GPU variable - quantized depth map data.
- double * [img_src_](#)

- CPU variable - image source data, values are from 0 to 1.*

 - double * [dmap_src_](#)
- CPU variable - depth map data, values are from 0 to 1.*

 - double * [depth_index_](#)
- CPU variable - quantized depth map data.*

 - int * [alpha_map_](#)
- CPU variable - calculated alpha map data, values are 0 or 1.*

 - double * [dmap_](#)
- CPU variable - physical distances of depth map.*

 - double [dstep_](#)
- the physical increment of each depth map layer.*

 - std::vector< double > [dlevel_](#)
- the physical value of all depth map layer.*

 - std::vector< double > [dlevel_transform_](#)
- transformed dlevel_ variable*

 - [Complex](#) * [U_complex_](#)
- CPU variable - the generated hologram before encoding.*

 - double * [u255_fringe_](#)
- the final hologram, used for writing the result image.*

 - [HologramParams](#) [params_](#)
- structure variable for hologram parameters*

 - std::string [SOURCE_FOLDER](#)
- input source folder - config file.*

 - std::string [IMAGE_PREFIX](#)
- the prefix of the input image file - config file.*

 - std::string [DEPTH_PREFIX](#)
- the prefix of the deptmap file - config file*

 - std::string [RESULT_FOLDER](#)
- the name of the result folder - config file*

 - std::string [RESULT_PREFIX](#)
- the prefix of the result file - config file*

 - bool [FLAG_STATIC_IMAGE](#)
- if true, the input image is static.*

 - uint [START_OF_FRAME_NUMBERING](#)
- the start frame number.*

 - uint [NUMBER_OF_FRAME](#)
- the total number of the frame.*

 - uint [NUMBER_OF_DIGIT_OF_FRAME_NUMBERING](#)
- the number of digit of frame number.*

 - int [Transform_Method_](#)
- transform method*

 - int [Propagation_Method_](#)
- propagation method - currently AngularSpectrum*

 - int [Encoding_Method_](#)
- encoding method - currently Symmetrization*

 - double [WAVELENGTH](#)
- wave length*

 - bool [FLAG_CHANGE_DEPTH_QUANTIZATION](#)
- if true, change the depth quantization from the default value.*

 - uint [DEFAULT_DEPTH_QUANTIZATION](#)
- default value of the depth quantization - 256*

- uint [NUMBER_OF_DEPTH_QUANTIZATION](#)
depth level of input depthmap.
- bool [RANDOM_PHASE](#)
If true, random phase is imposed on each depth layer.
- std::string [Simulation_Result_File_Prefix_](#)
reconstruction variable for testing
- int [test_pixel_number_scale_](#)
reconstruction variable for testing
- [vec2 Pixel_pitch_xy_](#)
reconstruction variable for testing
- [ivec2 SLM_pixel_number_xy_](#)
reconstruction variable for testing
- double [f_field_](#)
reconstruction variable for testing
- double [eye_length_](#)
reconstruction variable for testing
- double [eye_pupil_diameter_](#)
reconstruction variable for testing
- [vec2 eye_center_xy_](#)
reconstruction variable for testing
- double [focus_distance_](#)
reconstruction variable for testing
- int [sim_type_](#)
reconstruction variable for testing
- double [sim_from_](#)
reconstruction variable for testing
- double [sim_to_](#)
reconstruction variable for testing
- int [sim_step_num_](#)
reconstruction variable for testing
- double * [sim_final_](#)
reconstruction variable for testing
- [Complex](#) * [hh_complex_](#)
reconstruction variable for testing

9.3.1 Detailed Description

This is a main class for generating a digital hologram using depth map data. It is implemented on the CPU and GPU.

1. Read Config file. - to set all parameters needed for generating a hologram.
2. Initialize all variables. - memory allocation on the CPU and GPU.
3. Generate a digital hologram using depth map data.
4. For the testing purpose, reconstruct a image from the generated hologram.

Definition at line 130 of file HologramGenerator.h.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 HologramGenerator()

```
HologramGenerator::HologramGenerator ( )
```

Initialize variables.

Definition at line 15 of file HologramGenerator.cpp.

9.3.2.2 ~HologramGenerator()

```
HologramGenerator::~~HologramGenerator ( )
```

Definition at line 43 of file HologramGenerator.cpp.

9.3.3 Member Function Documentation

9.3.3.1 exponent_complex()

```
void HologramGenerator::exponent_complex (
    Complex * val ) [private]
```

Parameters

<i>val</i>	: input & ouput value
------------	-----------------------

See also

[Propagation_AngularSpectrum_CPU](#), [Calc_Holo_CPU](#)

Definition at line 351 of file HologramGenerator_CPU.cpp.

9.3.3.2 fftShift()

```
void HologramGenerator::fftShift (
    int nx,
    int ny,
    Complex * input,
    Complex * output ) [private]
```

Parameters

<i>nx</i>	: the number of column of the input data
<i>ny</i>	: the number of row of the input data
<i>input</i>	: input data variable
<i>output</i>	: output data variable

See also

[fftwShift](#)

Definition at line 332 of file HologramGenerator_CPU.cpp.

9.3.3.3 `fftwShift()`

```
void HologramGenerator::fftwShift (
    Complex * src,
    Complex * dst,
    fftw_complex * in,
    fftw_complex * out,
    int nx,
    int ny,
    int type,
    bool bNomarlized = false ) [private]
```

It is equivalent to Matlab code, `dst = ifftshift(fft2(fftshift(src)))`.

Parameters

<i>src</i>	: input data variable
<i>dst</i>	: output data variable
<i>in</i>	: input data pointer connected with FFTW plan
<i>out</i>	: ouput data pointer connected with FFTW plan
<i>nx</i>	: the number of column of the input data
<i>ny</i>	: the number of row of the input data
<i>type</i>	: If type == 1, forward FFT, if type == -1, backward FFT.
<i>bNomarlized</i>	: If bNomarlized == true, normalize the result after FFT.

See also

[Propagation_AngularSpectrum_CPU](#), [encoding_CPU](#)

Definition at line 285 of file HologramGenerator_CPU.cpp.

9.3.3.4 get_rand_phase_value()

```
void HologramGenerator::get_rand_phase_value (
    Complex & rand_phase_val ) [private]
```

If RANDOM_PHASE == 1, calculate a random phase value using random generator; otherwise, random phase value is 1.

Parameters

<i>rand_phase_val</i>	: Input & Ouput value.
-----------------------	------------------------

Definition at line 457 of file HologramGenerator.cpp.

9.3.3.5 get_shift_phase_value()

```
void HologramGenerator::get_shift_phase_value (
    Complex & shift_phase_val,
    int idx,
    ivec2 sig_location ) [private]
```

Parameters

<i>shift_phase_val</i>	: output variable.
<i>idx</i>	: the current pixel position.
<i>sig_location</i>	: signal location.

See also

[encoding_CPU](#)

Definition at line 367 of file HologramGenerator_CPU.cpp.

9.3.3.6 setMode()

```
void HologramGenerator::setMode (
    bool isCPU )
```

```
if isCPU_ == true
    CPU implementation
else
    GPU implementation
```

Parameters

<i>isCPU</i>	: the value for specifying whether the hologram generation method is implemented on the CPU or GPU
--------------	--

Definition at line 56 of file HologramGenerator.cpp.

9.3.4 Member Data Documentation

9.3.4.1 `alpha_map_`

```
int* HologramGenerator::alpha_map_ [private]
```

Definition at line 234 of file HologramGenerator.h.

9.3.4.2 `DEFAULT_DEPTH_QUANTIZATION`

```
uint HologramGenerator::DEFAULT_DEPTH_QUANTIZATION [private]
```

Definition at line 264 of file HologramGenerator.h.

9.3.4.3 `depth_index_`

```
double* HologramGenerator::depth_index_ [private]
```

Definition at line 233 of file HologramGenerator.h.

9.3.4.4 `depth_index_gpu_`

```
double* HologramGenerator::depth_index_gpu_ [private]
```

Definition at line 229 of file HologramGenerator.h.

9.3.4.5 `DEPTH_PREFIX`

```
std::string HologramGenerator::DEPTH_PREFIX [private]
```

Definition at line 249 of file HologramGenerator.h.

9.3.4.6 dimg_src_gpu_

```
unsigned char* HologramGenerator::dimg_src_gpu_ [private]
```

Definition at line 228 of file HologramGenerator.h.

9.3.4.7 dlevel_

```
std::vector<double> HologramGenerator::dlevel_ [private]
```

Definition at line 239 of file HologramGenerator.h.

9.3.4.8 dlevel_transform_

```
std::vector<double> HologramGenerator::dlevel_transform_ [private]
```

Definition at line 240 of file HologramGenerator.h.

9.3.4.9 dmap_

```
double* HologramGenerator::dmap_ [private]
```

Definition at line 236 of file HologramGenerator.h.

9.3.4.10 dmap_src_

```
double* HologramGenerator::dmap_src_ [private]
```

Definition at line 232 of file HologramGenerator.h.

9.3.4.11 dstep_

```
double HologramGenerator::dstep_ [private]
```

Definition at line 238 of file HologramGenerator.h.

9.3.4.12 Encoding_Method_

```
int HologramGenerator::Encoding_Method_ [private]
```

Definition at line 259 of file HologramGenerator.h.

9.3.4.13 eye_center_xy_

```
vec2 HologramGenerator::eye_center_xy_ [private]
```

Definition at line 277 of file HologramGenerator.h.

9.3.4.14 eye_length_

```
double HologramGenerator::eye_length_ [private]
```

Definition at line 275 of file HologramGenerator.h.

9.3.4.15 eye_pupil_diameter_

```
double HologramGenerator::eye_pupil_diameter_ [private]
```

Definition at line 276 of file HologramGenerator.h.

9.3.4.16 f_field_

```
double HologramGenerator::f_field_ [private]
```

Definition at line 274 of file HologramGenerator.h.

9.3.4.17 FLAG_CHANGE_DEPTH_QUANTIZATION

```
bool HologramGenerator::FLAG_CHANGE_DEPTH_QUANTIZATION [private]
```

Definition at line 263 of file HologramGenerator.h.

9.3.4.18 FLAG_STATIC_IMAGE

```
bool HologramGenerator::FLAG_STATIC_IMAGE [private]
```

Definition at line 252 of file HologramGenerator.h.

9.3.4.19 focus_distance_

```
double HologramGenerator::focus_distance_ [private]
```

Definition at line 278 of file HologramGenerator.h.

9.3.4.20 hh_complex_

```
Complex* HologramGenerator::hh_complex_ [private]
```

Definition at line 284 of file HologramGenerator.h.

9.3.4.21 IMAGE_PREFIX

```
std::string HologramGenerator::IMAGE_PREFIX [private]
```

Definition at line 248 of file HologramGenerator.h.

9.3.4.22 img_src_

```
double* HologramGenerator::img_src_ [private]
```

Definition at line 231 of file HologramGenerator.h.

9.3.4.23 img_src_gpu_

```
unsigned char* HologramGenerator::img_src_gpu_ [private]
```

Definition at line 227 of file HologramGenerator.h.

9.3.4.24 isCPU_

```
bool HologramGenerator::isCPU_ [private]
```

Definition at line 225 of file HologramGenerator.h.

9.3.4.25 NUMBER_OF_DEPTH_QUANTIZATION

```
uint HologramGenerator::NUMBER_OF_DEPTH_QUANTIZATION [private]
```

Definition at line 265 of file HologramGenerator.h.

9.3.4.26 NUMBER_OF_DIGIT_OF_FRAME_NUMBERING

```
uint HologramGenerator::NUMBER_OF_DIGIT_OF_FRAME_NUMBERING [private]
```

Definition at line 255 of file HologramGenerator.h.

9.3.4.27 NUMBER_OF_FRAME

```
uint HologramGenerator::NUMBER_OF_FRAME [private]
```

Definition at line 254 of file HologramGenerator.h.

9.3.4.28 params_

```
HologramParams HologramGenerator::params_ [private]
```

Definition at line 245 of file HologramGenerator.h.

9.3.4.29 Pixel_pitch_xy_

```
vec2 HologramGenerator::Pixel_pitch_xy_ [private]
```

Definition at line 272 of file HologramGenerator.h.

9.3.4.30 Propagation_Method_

```
int HologramGenerator::Propagation_Method_ [private]
```

Definition at line 258 of file HologramGenerator.h.

9.3.4.31 RANDOM_PHASE

```
bool HologramGenerator::RANDOM_PHASE [private]
```

Definition at line 266 of file HologramGenerator.h.

9.3.4.32 RESULT_FOLDER

```
std::string HologramGenerator::RESULT_FOLDER [private]
```

Definition at line 250 of file HologramGenerator.h.

9.3.4.33 RESULT_PREFIX

```
std::string HologramGenerator::RESULT_PREFIX [private]
```

Definition at line 251 of file HologramGenerator.h.

9.3.4.34 sim_final_

```
double* HologramGenerator::sim_final_ [private]
```

Definition at line 283 of file HologramGenerator.h.

9.3.4.35 sim_from_

```
double HologramGenerator::sim_from_ [private]
```

Definition at line 280 of file HologramGenerator.h.

9.3.4.36 `sim_step_num_`

```
int HologramGenerator::sim_step_num_ [private]
```

Definition at line 282 of file HologramGenerator.h.

9.3.4.37 `sim_to_`

```
double HologramGenerator::sim_to_ [private]
```

Definition at line 281 of file HologramGenerator.h.

9.3.4.38 `sim_type_`

```
int HologramGenerator::sim_type_ [private]
```

Definition at line 279 of file HologramGenerator.h.

9.3.4.39 `Simulation_Result_File_Prefix_`

```
std::string HologramGenerator::Simulation_Result_File_Prefix_ [private]
```

Definition at line 270 of file HologramGenerator.h.

9.3.4.40 `SLM_pixel_number_xy_`

```
ivec2 HologramGenerator::SLM_pixel_number_xy_ [private]
```

Definition at line 273 of file HologramGenerator.h.

9.3.4.41 `SOURCE_FOLDER`

```
std::string HologramGenerator::SOURCE_FOLDER [private]
```

Definition at line 247 of file HologramGenerator.h.

9.3.4.42 START_OF_FRAME_NUMBERING

```
uint HologramGenerator::START_OF_FRAME_NUMBERING [private]
```

Definition at line 253 of file HologramGenerator.h.

9.3.4.43 test_pixel_number_scale_

```
int HologramGenerator::test_pixel_number_scale_ [private]
```

Definition at line 271 of file HologramGenerator.h.

9.3.4.44 Transform_Method_

```
int HologramGenerator::Transform_Method_ [private]
```

Definition at line 257 of file HologramGenerator.h.

9.3.4.45 u255_fringe_

```
double* HologramGenerator::u255_fringe_ [private]
```

Definition at line 243 of file HologramGenerator.h.

9.3.4.46 U_complex_

```
Complex* HologramGenerator::U_complex_ [private]
```

Definition at line 242 of file HologramGenerator.h.

9.3.4.47 WAVELENGTH

```
double HologramGenerator::WAVELENGTH [private]
```

Definition at line 261 of file HologramGenerator.h.

The documentation for this class was generated from the following files:

- [Hologram/HologramGenerator.h](#)
- [Hologram/src/HologramGenerator.cpp](#)
- [Hologram/src/HologramGenerator_CPU.cpp](#)
- [Hologram/src/HologramGenerator_GPU.cpp](#)

9.4 HologramParams Struct Reference

Structure variable for hologram paramemters.

```
#include <HologramGenerator.h>
```

Public Attributes

- double [field_lens](#)
FIELD_LENS at config file.
- double [lambda](#)
WAVELENGTH at config file.
- double [k](#)
 $2 * \pi / \text{lambda}$
- [ivec2](#) [pn](#)
SLM_PIXEL_NUMBER_X & SLM_PIXEL_NUMBER_Y.
- [vec2](#) [pp](#)
SLM_PIXEL_PITCH_X & SLM_PIXEL_PITCH_Y.
- [vec2](#) [ss](#)
 $pn * pp$
- double [near_depthmap](#)
NEAR_OF_DEPTH_MAP at config file.
- double [far_depthmap](#)
FAR_OF_DEPTH_MAP at config file.
- uint [num_of_depth](#)
the number of depth level.
- `std::vector< int >` [render_depth](#)
Used when only few specific depth levels are rendered, usually for test purpose.

9.4.1 Detailed Description

This structure has all necessary parameters for generating a hologram. It is read from the configuration file, 'config↵_openholo.txt'.

Definition at line 100 of file HologramGenerator.h.

9.4.2 Member Data Documentation

9.4.2.1 [far_depthmap](#)

```
double HologramParams::far_depthmap
```

Definition at line 110 of file HologramGenerator.h.

9.4.2.2 field_lens

```
double HologramParams::field_lens
```

Definition at line 102 of file HologramGenerator.h.

9.4.2.3 k

```
double HologramParams::k
```

Definition at line 104 of file HologramGenerator.h.

9.4.2.4 lambda

```
double HologramParams::lambda
```

Definition at line 103 of file HologramGenerator.h.

9.4.2.5 near_depthmap

```
double HologramParams::near_depthmap
```

Definition at line 109 of file HologramGenerator.h.

9.4.2.6 num_of_depth

```
uint HologramParams::num_of_depth
```

```
if FLAG_CHANGE_DEPTH_QUANTIZATION == 0
    num_of_depth = DEFAULT_DEPTH_QUANTIZATION
else
    num_of_depth = NUMBER_OF_DEPTH_QUANTIZATION
```

Definition at line 112 of file HologramGenerator.h.

9.4.2.7 pn

`ivec2` HologramParams::pn

Definition at line 105 of file HologramGenerator.h.

9.4.2.8 pp

`vec2` HologramParams::pp

Definition at line 106 of file HologramGenerator.h.

9.4.2.9 render_depth

`std::vector<int>` HologramParams::render_depth

Definition at line 119 of file HologramGenerator.h.

9.4.2.10 ss

`vec2` HologramParams::ss

Definition at line 107 of file HologramGenerator.h.

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

- Hologram/[HologramGenerator.h](#)

9.5 graphics::ivec2 Struct Reference

structure for 2-dimensional integer vector and its arithmetic.

```
#include <ivec.h>
```

Public Member Functions

- [ivec2](#) ()
- [ivec2](#) (int a)
- [ivec2](#) (int v_1, int v_2)
- [ivec2](#) (const [ivec2](#) &a)
- [ivec2](#) & [operator=](#) (const [ivec2](#) &a)
- int & [operator\[\]](#) (int i)
- const int & [operator\[\]](#) (int i) const
- int & [operator\(\)](#) (int i)
- const int & [operator\(\)](#) (int i) const

Public Attributes

- int `v` [2]

Static Public Attributes

- static const int `n`

9.5.1 Detailed Description

Definition at line 14 of file ivec.h.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 ivec2() [1/4]

```
graphics::ivec2::ivec2 ( ) [inline]
```

Definition at line 18 of file ivec.h.

9.5.2.2 ivec2() [2/4]

```
graphics::ivec2::ivec2 (  
    int a ) [inline]
```

Definition at line 20 of file ivec.h.

9.5.2.3 ivec2() [3/4]

```
graphics::ivec2::ivec2 (  
    int v_1,  
    int v_2 ) [inline]
```

Definition at line 25 of file ivec.h.

9.5.2.4 `ivec2()` [4/4]

```
graphics::ivec2::ivec2 (  
    const ivec2 & a ) [inline]
```

Definition at line 30 of file `ivec.h`.

9.5.3 Member Function Documentation

9.5.3.1 `operator()` [1/2]

```
int& graphics::ivec2::operator() (  
    int i ) [inline]
```

Definition at line 44 of file `ivec.h`.

9.5.3.2 `operator()` [2/2]

```
const int& graphics::ivec2::operator() (  
    int i ) const [inline]
```

Definition at line 45 of file `ivec.h`.

9.5.3.3 `operator=()`

```
ivec2& graphics::ivec2::operator= (  
    const ivec2 & a ) [inline]
```

Definition at line 35 of file `ivec.h`.

9.5.3.4 `operator[]()` [1/2]

```
int& graphics::ivec2::operator[] (  
    int i ) [inline]
```

Definition at line 42 of file `ivec.h`.

9.5.3.5 operator[]() [2/2]

```
const int& graphics::ivec2::operator[] (
    int i ) const [inline]
```

Definition at line 43 of file ivec.h.

9.5.4 Member Data Documentation

9.5.4.1 n

```
const int graphics::ivec2::n [static]
```

Definition at line 16 of file ivec.h.

9.5.4.2 v

```
int graphics::ivec2::v[2]
```

Definition at line 15 of file ivec.h.

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

- graphics/[ivec.h](#)

9.6 graphics::ivec3 Struct Reference

structure for 3-dimensional integer vector and its arithmetic.

```
#include <ivec.h>
```

Public Member Functions

- [ivec3](#) ()
- [ivec3](#) (int a)
- [ivec3](#) (int v_1, int v_2, int v_3)
- [ivec3](#) (const [ivec3](#) &a)
- [ivec3](#) & [operator=](#) (const [ivec3](#) &a)
- int & [operator\[\]](#) (int i)
- const int & [operator\[\]](#) (int i) const
- int & [operator\(\)](#) (int i)
- const int & [operator\(\)](#) (int i) const

Public Attributes

- int `v` [3]

Static Public Attributes

- static const int `n`

9.6.1 Detailed Description

Definition at line 331 of file `ivec.h`.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 `ivec3()` [1/4]

```
graphics::ivec3::ivec3 ( ) [inline]
```

Definition at line 335 of file `ivec.h`.

9.6.2.2 `ivec3()` [2/4]

```
graphics::ivec3::ivec3 (  
    int a ) [inline]
```

Definition at line 337 of file `ivec.h`.

9.6.2.3 `ivec3()` [3/4]

```
graphics::ivec3::ivec3 (  
    int v_1,  
    int v_2,  
    int v_3 ) [inline]
```

Definition at line 342 of file `ivec.h`.

9.6.2.4 ivec3() [4/4]

```
graphics::ivec3::ivec3 (  
    const ivec3 & a ) [inline]
```

Definition at line 347 of file ivec.h.

9.6.3 Member Function Documentation

9.6.3.1 operator() [1/2]

```
int& graphics::ivec3::operator() (  
    int i ) [inline]
```

Definition at line 361 of file ivec.h.

9.6.3.2 operator() [2/2]

```
const int& graphics::ivec3::operator() (  
    int i ) const [inline]
```

Definition at line 362 of file ivec.h.

9.6.3.3 operator=()

```
ivec3& graphics::ivec3::operator= (  
    const ivec3 & a ) [inline]
```

Definition at line 352 of file ivec.h.

9.6.3.4 operator[]() [1/2]

```
int& graphics::ivec3::operator[] (  
    int i ) [inline]
```

Definition at line 359 of file ivec.h.

9.6.3.5 operator[]() [2/2]

```
const int& graphics::ivec3::operator[] (
    int i ) const [inline]
```

Definition at line 360 of file ivec.h.

9.6.4 Member Data Documentation

9.6.4.1 n

```
const int graphics::ivec3::n [static]
```

Definition at line 333 of file ivec.h.

9.6.4.2 v

```
int graphics::ivec3::v[3]
```

Definition at line 332 of file ivec.h.

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

- [graphics/ivec.h](#)

9.7 graphics::ivec4 Struct Reference

structure for 4-dimensional integer vector and its arithmetic.

```
#include <ivec.h>
```

Public Member Functions

- [ivec4](#) ()
- [ivec4](#) (int a)
- [ivec4](#) (int v_1, int v_2, int v_3, int v_4)
- [ivec4](#) (const [ivec4](#) &a)
- [ivec4](#) & [operator=](#) (const [ivec4](#) &a)
- int & [operator\[\]](#) (int i)
- const int & [operator\[\]](#) (int i) const
- int & [operator\(\)](#) (int i)
- const int & [operator\(\)](#) (int i) const

Public Attributes

- int `v`[4]

Static Public Attributes

- static const int `n`

9.7.1 Detailed Description

Definition at line 643 of file `ivec.h`.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 `ivec4()` [1/4]

```
graphics::ivec4::ivec4 ( ) [inline]
```

Definition at line 647 of file `ivec.h`.

9.7.2.2 `ivec4()` [2/4]

```
graphics::ivec4::ivec4 (  
    int a ) [inline]
```

Definition at line 649 of file `ivec.h`.

9.7.2.3 `ivec4()` [3/4]

```
graphics::ivec4::ivec4 (  
    int v_1,  
    int v_2,  
    int v_3,  
    int v_4 ) [inline]
```

Definition at line 654 of file `ivec.h`.

9.7.2.4 `ivec4()` [4/4]

```
graphics::ivec4::ivec4 (  
    const ivec4 & a ) [inline]
```

Definition at line 659 of file `ivec.h`.

9.7.3 Member Function Documentation

9.7.3.1 `operator()` [1/2]

```
int& graphics::ivec4::operator() (  
    int i ) [inline]
```

Definition at line 673 of file `ivec.h`.

9.7.3.2 `operator()` [2/2]

```
const int& graphics::ivec4::operator() (  
    int i ) const [inline]
```

Definition at line 674 of file `ivec.h`.

9.7.3.3 `operator=()`

```
ivec4& graphics::ivec4::operator= (  
    const ivec4 & a ) [inline]
```

Definition at line 664 of file `ivec.h`.

9.7.3.4 `operator[]()` [1/2]

```
int& graphics::ivec4::operator[] (  
    int i ) [inline]
```

Definition at line 671 of file `ivec.h`.

9.7.3.5 operator[]() [2/2]

```
const int& graphics::ivec4::operator[] (
    int i ) const [inline]
```

Definition at line 672 of file ivec.h.

9.7.4 Member Data Documentation

9.7.4.1 n

```
const int graphics::ivec4::n [static]
```

Definition at line 645 of file ivec.h.

9.7.4.2 v

```
int graphics::ivec4::v[4]
```

Definition at line 644 of file ivec.h.

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

- [graphics/ivec.h](#)

9.8 graphics::vec2 Struct Reference

structure for 2-dimensional real type vector and its arithmetic.

```
#include <vec.h>
```

Public Member Functions

- [vec2](#) ()
- [vec2](#) (real a)
- [vec2](#) (real v_1, real v_2)
- [vec2](#) (const [ivec2](#) &a)
- [vec2](#) (const [vec2](#) &a)
- [vec2](#) & [operator=](#) (const [vec2](#) &a)
- [real](#) & [operator\[\]](#) (int i)
- const [real](#) & [operator\[\]](#) (int i) const
- [real](#) & [operator\(\)](#) (int i)
- const [real](#) & [operator\(\)](#) (int i) const
- bool [unit](#) ()
- [real](#) [length](#) () const
- bool [is_zero](#) () const
- bool [is_tiny](#) (real tiny_tol=[epsilon](#)) const
- int [is_parallel](#) (const [vec2](#) &, real=[angle_tolerance](#)) const
- bool [is_perpendicular](#) (const [vec2](#) &, real=[angle_tolerance](#)) const
- bool [perpendicular](#) (const [vec2](#) &)
- bool [perpendicular](#) (const [vec2](#) &, const [vec2](#) &)

Public Attributes

- `real v [2]`

Static Public Attributes

- `static const int n = 2`

9.8.1 Detailed Description

Definition at line 22 of file `vec.h`.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 `vec2()` [1/5]

```
graphics::vec2::vec2 ( ) [inline]
```

Definition at line 26 of file `vec.h`.

9.8.2.2 `vec2()` [2/5]

```
graphics::vec2::vec2 (
    real a ) [inline]
```

Definition at line 27 of file `vec.h`.

9.8.2.3 `vec2()` [3/5]

```
graphics::vec2::vec2 (
    real v_1,
    real v_2 ) [inline]
```

Definition at line 32 of file `vec.h`.

9.8.2.4 vec2() [4/5]

```
graphics::vec2::vec2 (
    const ivec2 & a ) [inline]
```

Definition at line 37 of file vec.h.

9.8.2.5 vec2() [5/5]

```
graphics::vec2::vec2 (
    const vec2 & a ) [inline]
```

Definition at line 42 of file vec.h.

9.8.3 Member Function Documentation

9.8.3.1 is_parallel()

```
int graphics::vec2::is_parallel (
    const vec2 & vv,
    real angle_tolerance = angle\_tolerance ) const
```

Definition at line 22 of file vec.cpp.

9.8.3.2 is_perpendicular()

```
bool graphics::vec2::is_perpendicular (
    const vec2 & vv,
    real angle_tolerance = angle\_tolerance ) const
```

Definition at line 44 of file vec.cpp.

9.8.3.3 is_tiny()

```
bool graphics::vec2::is_tiny (
    real tiny_tol = epsilon ) const [inline]
```

Definition at line 62 of file vec.h.

9.8.3.4 is_zero()

```
bool graphics::vec2::is_zero ( ) const [inline]
```

Definition at line 61 of file vec.h.

9.8.3.5 length()

```
real graphics::vec2::length ( ) const
```

Definition at line 17 of file vec.cpp.

9.8.3.6 operator() [1/2]

```
real& graphics::vec2::operator() (
    int i ) [inline]
```

Definition at line 55 of file vec.h.

9.8.3.7 operator() [2/2]

```
const real& graphics::vec2::operator() (
    int i ) const [inline]
```

Definition at line 56 of file vec.h.

9.8.3.8 operator=()

```
vec2& graphics::vec2::operator= (
    const vec2 & a ) [inline]
```

Definition at line 47 of file vec.h.

9.8.3.9 operator[] [1/2]

```
real& graphics::vec2::operator[] (
    int i ) [inline]
```

Definition at line 53 of file vec.h.

9.8.3.10 operator[]() [2/2]

```
const real& graphics::vec2::operator[] (
    int i ) const [inline]
```

Definition at line 54 of file vec.h.

9.8.3.11 perpendicular() [1/2]

```
bool graphics::vec2::perpendicular (
    const vec2 & vv )
```

Definition at line 62 of file vec.cpp.

9.8.3.12 perpendicular() [2/2]

```
bool graphics::vec2::perpendicular (
    const vec2 & p,
    const vec2 & q )
```

Definition at line 73 of file vec.cpp.

9.8.3.13 unit()

```
bool graphics::vec2::unit ( )
```

Definition at line 8 of file vec.cpp.

9.8.4 Member Data Documentation

9.8.4.1 n

```
const int graphics::vec2::n = 2 [static]
```

Definition at line 24 of file vec.h.

9.8.4.2 v

```
real graphics::vec2::v[2]
```

Definition at line 23 of file vec.h.

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

- [graphics/vec.h](#)
- [graphics/src/vec.cpp](#)

9.9 graphics::vec3 Struct Reference

structure for 3-dimensional real type vector and its arithmetic.

```
#include <vec.h>
```

Public Member Functions

- [vec3](#) ()
- [vec3](#) (real a)
- [vec3](#) (real v_1, real v_2, real v_3)
- [vec3](#) (const [ivec3](#) &a)
- [vec3](#) (const [vec3](#) &a)
- [vec3](#) & [operator=](#) (const [vec3](#) &a)
- [real](#) & [operator\[\]](#) (int i)
- const [real](#) & [operator\[\]](#) (int i) const
- [real](#) & [operator\(\)](#) (int i)
- const [real](#) & [operator\(\)](#) (int i) const
- bool [is_zero](#) () const
- bool [is_tiny](#) (real tiny_tol=[epsilon](#)) const
- bool [unit](#) ()
- [real](#) [length](#) () const
- int [is_parallel](#) (const [vec3](#) &, real=[angle_tolerance](#)) const
- bool [is_perpendicular](#) (const [vec3](#) &, real=[angle_tolerance](#)) const
- bool [perpendicular](#) (const [vec3](#) &)
- bool [perpendicular](#) (const [vec3](#) &, const [vec3](#) &, const [vec3](#) &)

Public Attributes

- [real](#) v [3]

Static Public Attributes

- static const int n = 3

9.9.1 Detailed Description

Definition at line 444 of file vec.h.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 vec3() [1/5]

```
graphics::vec3::vec3 ( ) [inline]
```

Definition at line 448 of file vec.h.

9.9.2.2 vec3() [2/5]

```
graphics::vec3::vec3 (  
    real a ) [inline]
```

Definition at line 449 of file vec.h.

9.9.2.3 vec3() [3/5]

```
graphics::vec3::vec3 (  
    real v_1,  
    real v_2,  
    real v_3 ) [inline]
```

Definition at line 454 of file vec.h.

9.9.2.4 vec3() [4/5]

```
graphics::vec3::vec3 (  
    const ivec3 & a ) [inline]
```

Definition at line 459 of file vec.h.

9.9.2.5 `vec3()` [5/5]

```
graphics::vec3::vec3 (
    const vec3 & a ) [inline]
```

Definition at line 464 of file `vec.h`.

9.9.3 Member Function Documentation

9.9.3.1 `is_parallel()`

```
int graphics::vec3::is_parallel (
    const vec3 & vv,
    real angle_tolerance = angle_tolerance ) const
```

Definition at line 142 of file `vec.cpp`.

9.9.3.2 `is_perpendicular()`

```
bool graphics::vec3::is_perpendicular (
    const vec3 & vv,
    real angle_tolerance = angle_tolerance ) const
```

Definition at line 164 of file `vec.cpp`.

9.9.3.3 `is_tiny()`

```
bool graphics::vec3::is_tiny (
    real tiny_tol = epsilon ) const [inline]
```

Definition at line 481 of file `vec.h`.

9.9.3.4 `is_zero()`

```
bool graphics::vec3::is_zero ( ) const [inline]
```

Definition at line 480 of file `vec.h`.

9.9.3.5 length()

```
real graphics::vec3::length ( ) const
```

Definition at line 137 of file vec.cpp.

9.9.3.6 operator>() [1/2]

```
real& graphics::vec3::operator() (
    int i ) [inline]
```

Definition at line 477 of file vec.h.

9.9.3.7 operator>() [2/2]

```
const real& graphics::vec3::operator() (
    int i ) const [inline]
```

Definition at line 478 of file vec.h.

9.9.3.8 operator=()

```
vec3& graphics::vec3::operator= (
    const vec3 & a ) [inline]
```

Definition at line 469 of file vec.h.

9.9.3.9 operator[]() [1/2]

```
real& graphics::vec3::operator[] (
    int i ) [inline]
```

Definition at line 475 of file vec.h.

9.9.3.10 operator[]() [2/2]

```
const real& graphics::vec3::operator[] (
    int i ) const [inline]
```

Definition at line 476 of file vec.h.

9.9.3.11 perpendicular() [1/2]

```
bool graphics::vec3::perpendicular (
    const vec3 & vv )
```

Definition at line 182 of file [vec.cpp](#).

9.9.3.12 perpendicular() [2/2]

```
bool graphics::vec3::perpendicular (
    const vec3 & P0,
    const vec3 & P1,
    const vec3 & P2 )
```

Definition at line 244 of file [vec.cpp](#).

9.9.3.13 unit()

```
bool graphics::vec3::unit ( )
```

Definition at line 128 of file [vec.cpp](#).

9.9.4 Member Data Documentation

9.9.4.1 n

```
const int graphics::vec3::n = 3 [static]
```

Definition at line 446 of file [vec.h](#).

9.9.4.2 v

```
real graphics::vec3::v[3]
```

Definition at line 445 of file [vec.h](#).

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

- [graphics/vec.h](#)
- [graphics/src/vec.cpp](#)

9.10 graphics::vec4 Struct Reference

structure for 4-dimensional real type vector and its arithmetic.

```
#include <vec.h>
```

Public Member Functions

- [vec4](#) ()
- [vec4](#) ([real](#) a)
- [vec4](#) ([real](#) v_1, [real](#) v_2, [real](#) v_3, [real](#) v_4)
- [vec4](#) (const [ivec4](#) &a)
- [vec4](#) (const [vec4](#) &a)
- [vec4](#) & [operator=](#) (const [vec4](#) &a)
- [real](#) & [operator\[\]](#) (int i)
- const [real](#) & [operator\[\]](#) (int i) const
- [real](#) & [operator\(\)](#) (int i)
- const [real](#) & [operator\(\)](#) (int i) const
- bool [is_zero](#) () const
- bool [is_tiny](#) ([real](#) tiny_tol=[epsilon](#)) const
- bool [unit](#) ()
- [real](#) [length](#) () const

Public Attributes

- [real](#) v [4]

Static Public Attributes

- static const int n = 4

9.10.1 Detailed Description

Definition at line 864 of file vec.h.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 [vec4\(\)](#) [1/5]

```
graphics::vec4::vec4 ( ) [inline]
```

Definition at line 868 of file vec.h.

9.10.2.2 `vec4()` [2/5]

```
graphics::vec4::vec4 (
    real a ) [inline]
```

Definition at line 869 of file `vec.h`.

9.10.2.3 `vec4()` [3/5]

```
graphics::vec4::vec4 (
    real v_1,
    real v_2,
    real v_3,
    real v_4 ) [inline]
```

Definition at line 874 of file `vec.h`.

9.10.2.4 `vec4()` [4/5]

```
graphics::vec4::vec4 (
    const ivec4 & a ) [inline]
```

Definition at line 879 of file `vec.h`.

9.10.2.5 `vec4()` [5/5]

```
graphics::vec4::vec4 (
    const vec4 & a ) [inline]
```

Definition at line 884 of file `vec.h`.

9.10.3 Member Function Documentation

9.10.3.1 `is_tiny()`

```
bool graphics::vec4::is_tiny (
    real tiny_tol = epsilon ) const [inline]
```

Definition at line 901 of file `vec.h`.

9.10.3.2 is_zero()

```
bool graphics::vec4::is_zero ( ) const [inline]
```

Definition at line 900 of file vec.h.

9.10.3.3 length()

```
real graphics::vec4::length ( ) const
```

Definition at line 353 of file vec.cpp.

9.10.3.4 operator>() [1/2]

```
real& graphics::vec4::operator() (
    int i ) [inline]
```

Definition at line 897 of file vec.h.

9.10.3.5 operator>() [2/2]

```
const real& graphics::vec4::operator() (
    int i ) const [inline]
```

Definition at line 898 of file vec.h.

9.10.3.6 operator=()

```
vec4& graphics::vec4::operator= (
    const vec4 & a ) [inline]
```

Definition at line 889 of file vec.h.

9.10.3.7 operator[]() [1/2]

```
real& graphics::vec4::operator[] (
    int i ) [inline]
```

Definition at line 895 of file vec.h.

9.10.3.8 operator[]() [2/2]

```
const real& graphics::vec4::operator[] (
    int i ) const [inline]
```

Definition at line 896 of file vec.h.

9.10.3.9 unit()

```
bool graphics::vec4::unit ( )
```

Definition at line 344 of file vec.cpp.

9.10.4 Member Data Documentation

9.10.4.1 n

```
const int graphics::vec4::n = 4 [static]
```

Definition at line 866 of file vec.h.

9.10.4.2 v

```
real graphics::vec4::v[4]
```

Definition at line 865 of file vec.h.

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

- graphics/[vec.h](#)
- graphics/src/[vec.cpp](#)

Chapter 10

File Documentation

10.1 graphics/complex.h File Reference

```
#include <iostream>
#include <cmath>
```

Classes

- class [Complex](#)
class for the complex number and its arithmetic.

Variables

- const double [PI](#) = 3.141592653589793238462643383279502884197169399375105820974944592308
- const double [TWO_PI](#) = 2.0*[PI](#)

10.1.1 Variable Documentation

10.1.1.1 [PI](#)

```
const double PI = 3.141592653589793238462643383279502884197169399375105820974944592308
```

Definition at line 16 of file complex.h.

10.1.1.2 [TWO_PI](#)

```
const double TWO_PI = 2.0*PI
```

Definition at line 17 of file complex.h.

10.2 graphics/epsilon.h File Reference

```
#include "graphics/real.h"
```

Namespaces

- [graphics](#)

Functions

- void [graphics::set_u_epsilon](#) (real a)
- void [graphics::reset_u_epsilon](#) ()
- void [graphics::set_zero_epsilon](#) (real a)
- void [graphics::reset_zero_epsilon](#) ()
- int [graphics::apx_equal](#) (real x, real y)
- int [graphics::apx_equal](#) (real x, real y, real eps)

Variables

- [real graphics::epsilon](#) = 1.0e-8
- [real graphics::user_epsilon](#) = 1.0e-8
- [real graphics::intersection_epsilon](#) = 1e-6
- [real graphics::sqrt_epsilon](#) = 1.490116119385000000e-8
- [real graphics::unset_value](#) = -1.23432101234321e+308
- [real graphics::zero_tolerance](#) = 1.0e-12
- [real graphics::angle_tolerance](#) = [M_PI](#)/180.0
- [real graphics::zero_epsilon](#) = 1.0e-12

10.3 graphics/ivec.h File Reference

```
#include <stdio.h>
```

Classes

- struct [graphics::ivec2](#)
structure for 2-dimensional integer vector and its arithmetic.
- struct [graphics::ivec3](#)
structure for 3-dimensional integer vector and its arithmetic.
- struct [graphics::ivec4](#)
structure for 4-dimensional integer vector and its arithmetic.

Namespaces

- [graphics](#)

Functions

- `ivec2 graphics::operator+` (const ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator+` (int a, const ivec2 &b)
- `ivec2 graphics::operator+` (const ivec2 &a, int b)
- `ivec2 graphics::operator-` (const ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator-` (int a, const ivec2 &b)
- `ivec2 graphics::operator-` (const ivec2 &a, int b)
- `ivec2 graphics::operator*` (const ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator*` (int a, const ivec2 &b)
- `ivec2 graphics::operator*` (const ivec2 &a, int b)
- `ivec2 graphics::operator+=` (ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator+=` (ivec2 &a, int b)
- `ivec2 graphics::operator-=` (ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator-=` (ivec2 &a, int b)
- `ivec2 graphics::operator*=` (ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator*=` (ivec2 &a, int b)
- `int graphics::operator==` (const ivec2 &a, const ivec2 &b)
- `int graphics::operator==` (int a, const ivec2 &b)
- `int graphics::operator==` (const ivec2 &a, int b)
- `int graphics::operator<` (const ivec2 &a, const ivec2 &b)
- `int graphics::operator<` (int a, const ivec2 &b)
- `int graphics::operator<` (const ivec2 &a, int b)
- `int graphics::operator<=` (const ivec2 &a, const ivec2 &b)
- `int graphics::operator<=` (int a, const ivec2 &b)
- `int graphics::operator<=` (const ivec2 &a, int b)
- `int graphics::operator>` (const ivec2 &a, const ivec2 &b)
- `int graphics::operator>` (int a, const ivec2 &b)
- `int graphics::operator>` (const ivec2 &a, int b)
- `int graphics::operator>=` (const ivec2 &a, const ivec2 &b)
- `int graphics::operator>=` (int a, const ivec2 &b)
- `int graphics::operator>=` (const ivec2 &a, int b)
- `int graphics::operator!=` (const ivec2 &a, const ivec2 &b)
- `ivec2 graphics::operator-` (const ivec2 &a)
- `ivec3 graphics::operator+` (const ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator+` (int a, const ivec3 &b)
- `ivec3 graphics::operator+` (const ivec3 &a, int b)
- `ivec3 graphics::operator-` (const ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator-` (int a, const ivec3 &b)
- `ivec3 graphics::operator-` (const ivec3 &a, int b)
- `ivec3 graphics::operator*` (const ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator*` (int a, const ivec3 &b)
- `ivec3 graphics::operator*` (const ivec3 &a, int b)
- `ivec3 graphics::operator+=` (ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator+=` (ivec3 &a, int b)
- `ivec3 graphics::operator-=` (ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator-=` (ivec3 &a, int b)
- `ivec3 graphics::operator*=` (ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator*=` (ivec3 &a, int b)
- `int graphics::operator==` (const ivec3 &a, const ivec3 &b)
- `int graphics::operator==` (int a, const ivec3 &b)
- `int graphics::operator==` (const ivec3 &a, int b)
- `int graphics::operator<` (const ivec3 &a, const ivec3 &b)
- `int graphics::operator<` (int a, const ivec3 &b)
- `int graphics::operator<` (const ivec3 &a, int b)

- `int graphics::operator<=` (const ivec3 &a, const ivec3 &b)
- `int graphics::operator<=` (int a, const ivec3 &b)
- `int graphics::operator<=` (const ivec3 &a, int b)
- `int graphics::operator>` (const ivec3 &a, const ivec3 &b)
- `int graphics::operator>` (int a, const ivec3 &b)
- `int graphics::operator>` (const ivec3 &a, int b)
- `int graphics::operator>=` (const ivec3 &a, const ivec3 &b)
- `int graphics::operator>=` (int a, const ivec3 &b)
- `int graphics::operator>=` (const ivec3 &a, int b)
- `int graphics::operator!=` (const ivec3 &a, const ivec3 &b)
- `ivec3 graphics::operator-` (const ivec3 &a)
- `ivec4 graphics::operator+` (const ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator+` (int a, const ivec4 &b)
- `ivec4 graphics::operator+` (const ivec4 &a, int b)
- `ivec4 graphics::operator-` (const ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator-` (int a, const ivec4 &b)
- `ivec4 graphics::operator-` (const ivec4 &a, int b)
- `ivec4 graphics::operator*` (const ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator*` (int a, const ivec4 &b)
- `ivec4 graphics::operator*` (const ivec4 &a, int b)
- `ivec4 graphics::operator+=` (ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator+=` (ivec4 &a, int b)
- `ivec4 graphics::operator-=` (ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator-=` (ivec4 &a, int b)
- `ivec4 graphics::operator*=` (ivec4 &a, const ivec4 &b)
- `ivec4 graphics::operator*=` (ivec4 &a, int b)
- `int graphics::operator==` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator==` (int a, const ivec4 &b)
- `int graphics::operator==` (const ivec4 &a, int b)
- `int graphics::operator<` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator<` (int a, const ivec4 &b)
- `int graphics::operator<` (const ivec4 &a, int b)
- `int graphics::operator<=` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator<=` (int a, const ivec4 &b)
- `int graphics::operator<=` (const ivec4 &a, int b)
- `int graphics::operator>` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator>` (int a, const ivec4 &b)
- `int graphics::operator>` (const ivec4 &a, int b)
- `int graphics::operator>=` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator!=` (const ivec4 &a, const ivec4 &b)
- `int graphics::operator>=` (int a, const ivec4 &b)
- `int graphics::operator>=` (const ivec4 &a, int b)
- `ivec4 graphics::operator-` (const ivec4 &a)

10.4 graphics/real.h File Reference

Namespaces

- `graphics`

Macros

- `#define REAL_T_IS_FLOAT 1`
- `#define _MAXFLOAT ((float)3.40282347e+38)`
- `#define _MAXDOUBLE ((double)1.7976931348623158e+308)`
- `#define _MINFLOAT ((float)1.17549435e-38)`
- `#define _MINDOUBLE ((double)2.2250738585072014e-308)`
- `#define M_PI 3.141592653589793238462643383279502884197169399375105820974944592308`
- `#define MINREAL _MINDOUBLE;`
- `#define MAXREAL _MAXDOUBLE;`

Typedefs

- `typedef double real`
- `typedef float real_t`

10.4.1 Macro Definition Documentation

10.4.1.1 _MAXDOUBLE

```
#define _MAXDOUBLE ((double)1.7976931348623158e+308)
```

Definition at line 16 of file real.h.

10.4.1.2 _MAXFLOAT

```
#define _MAXFLOAT ((float)3.40282347e+38)
```

Definition at line 12 of file real.h.

10.4.1.3 _MINDOUBLE

```
#define _MINDOUBLE ((double)2.2250738585072014e-308)
```

Definition at line 20 of file real.h.

10.4.1.4 _MINFLOAT

```
#define _MINFLOAT ((float)1.17549435e-38)
```

Definition at line 19 of file real.h.

10.4.1.5 M_PI

```
#define M_PI 3.141592653589793238462643383279502884197169399375105820974944592308
```

Definition at line 23 of file real.h.

10.4.1.6 MAXREAL

```
#define MAXREAL _MAXDOUBLE;
```

Definition at line 27 of file real.h.

10.4.1.7 MINREAL

```
#define MINREAL _MINDOUBLE;
```

Definition at line 26 of file real.h.

10.4.1.8 REAL_T_IS_FLOAT

```
#define REAL_T_IS_FLOAT 1
```

Definition at line 7 of file real.h.

10.4.2 Typedef Documentation

10.4.2.1 real

```
typedef double real
```

Definition at line 4 of file real.h.

10.4.2.2 real_t

```
typedef float real_t
```

Definition at line 5 of file real.h.

10.5 graphics/src/epsilon.cpp File Reference

```
#include "graphics/epsilon.h"  
#include <math.h>  
#include "graphics/sys.h"
```

Namespaces

- [graphics](#)

Functions

- void [graphics::set_u_epsilon](#) (real a)
- void [graphics::reset_u_epsilon](#) ()
- void [graphics::set_zero_epsilon](#) (real a)
- void [graphics::reset_zero_epsilon](#) ()
- int [graphics::apx_equal](#) (real x, real y)
- int [graphics::apx_equal](#) (real x, real y, real eps)

Variables

- real [graphics::save_zero_epsilon](#) = 1.0e-12

10.6 graphics/src/sys.cpp File Reference

```
#include "graphics/sys.h"  
#include <stdarg.h>  
#include <stdio.h>
```

Functions

- void [file_log](#) (const char *fmt,...)
- FILE * [file_read_open](#) (const WChar *fname)
- FILE * [file_write_open](#) (const WChar *fname)
- FILE * [file_read_open](#) (const WChar *fname, const WChar *mode)
- FILE * [file_write_open](#) (const WChar *fname, const WChar *mode)
- WChar * [string_cpy](#) (WChar *dest, const WChar *src)
- WChar * [string_cat](#) (WChar *dest, const WChar *src)
- int [string_cmp](#) (const WChar *str1, const WChar *str2)

Variables

- static FILE * `fp`

10.6.1 Function Documentation

10.6.1.1 `file_log()`

```
void file_log (  
    const char * fmt,  
    ... )
```

Definition at line 7 of file `sys.cpp`.

10.6.1.2 `file_read_open()` [1/2]

```
FILE* file_read_open (  
    const WChar * fname )
```

Definition at line 38 of file `sys.cpp`.

10.6.1.3 `file_read_open()` [2/2]

```
FILE* file_read_open (  
    const WChar * fname,  
    const WChar * mode )
```

Definition at line 56 of file `sys.cpp`.

10.6.1.4 `file_write_open()` [1/2]

```
FILE* file_write_open (  
    const WChar * fname )
```

Definition at line 47 of file `sys.cpp`.

10.6.1.5 file_write_open() [2/2]

```
FILE* file_write_open (
    const WChar * fname,
    const WChar * mode )
```

Definition at line 65 of file sys.cpp.

10.6.1.6 string_cat()

```
WChar* string_cat (
    WChar * dest,
    const WChar * src )
```

Definition at line 83 of file sys.cpp.

10.6.1.7 string_cmp()

```
int string_cmp (
    const WChar * str1,
    const WChar * str2 )
```

Definition at line 92 of file sys.cpp.

10.6.1.8 string_cpy()

```
WChar* string_cpy (
    WChar * dest,
    const WChar * src )
```

Definition at line 74 of file sys.cpp.

10.6.2 Variable Documentation

10.6.2.1 fp

```
FILE* fp [static]
```

Definition at line 5 of file sys.cpp.

10.7 graphics/src/vec.cpp File Reference

```
#include "graphics/vec.h"
```

Namespaces

- [graphics](#)

Functions

- void [graphics::store](#) (FILE *[fp](#), const vec2 &[v](#))
- int [graphics::scan](#) (FILE *[fp](#), const vec2 &[v](#))
- int [graphics::apx_equal](#) (const vec2 &[a](#), const vec2 &[b](#))
- int [graphics::apx_equal](#) (const vec2 &[a](#), const vec2 &[b](#), [real](#) [eps](#))
- void [graphics::store](#) (FILE *[fp](#), const vec3 &[v](#))
- int [graphics::scan](#) (FILE *[fp](#), const vec3 &[v](#))
- int [graphics::apx_equal](#) (const vec3 &[a](#), const vec3 &[b](#))
- int [graphics::apx_equal](#) (const vec3 &[a](#), const vec3 &[b](#), [real](#) [eps](#))
- void [graphics::store](#) (FILE *[fp](#), const vec4 &[v](#))
- int [graphics::scan](#) (FILE *[fp](#), const vec4 &[v](#))
- int [graphics::apx_equal](#) (const vec4 &[a](#), const vec4 &[b](#))
- int [graphics::apx_equal](#) (const vec4 &[a](#), const vec4 &[b](#), [real](#) [eps](#))
- vec3 [graphics::cross](#) (const vec3 &[a](#), const vec3 &[b](#))

10.8 graphics/sys.h File Reference

```
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/types.h>
```

Macros

- `#define FLOG fprintf`

Functions

- FILE * [file_read_open](#) (const WChar *[fname](#))
- FILE * [file_write_open](#) (const WChar *[fname](#))
- FILE * [file_read_open](#) (const WChar *[fname](#), const WChar *[mode](#))
- FILE * [file_write_open](#) (const WChar *[fname](#), const WChar *[mode](#))
- WChar * [string_cpy](#) (WChar *[dest](#), const WChar *[src](#))
- WChar * [string_cat](#) (WChar *[dest](#), const WChar *[src](#))
- int [string_cmp](#) (const WChar *[str1](#), const WChar *[str2](#))

10.8.1 Macro Definition Documentation

10.8.1.1 FLOG

```
#define FLOG fprintf
```

Definition at line 73 of file sys.h.

10.8.2 Function Documentation

10.8.2.1 file_read_open() [1/2]

```
FILE* file_read_open (  
    const WChar * fname )
```

Definition at line 38 of file sys.cpp.

10.8.2.2 file_read_open() [2/2]

```
FILE* file_read_open (  
    const WChar * fname,  
    const WChar * mode )
```

Definition at line 56 of file sys.cpp.

10.8.2.3 file_write_open() [1/2]

```
FILE* file_write_open (  
    const WChar * fname )
```

Definition at line 47 of file sys.cpp.

10.8.2.4 file_write_open() [2/2]

```
FILE* file_write_open (  
    const WChar * fname,  
    const WChar * mode )
```

Definition at line 65 of file sys.cpp.

10.8.2.5 string_cat()

```
WChar* string_cat (
    WChar * dest,
    const WChar * src )
```

Definition at line 83 of file sys.cpp.

10.8.2.6 string_cmp()

```
int string_cmp (
    const WChar * str1,
    const WChar * str2 )
```

Definition at line 92 of file sys.cpp.

10.8.2.7 string_cpy()

```
WChar* string_cpy (
    WChar * dest,
    const WChar * src )
```

Definition at line 74 of file sys.cpp.

10.9 graphics/vec.h File Reference

```
#include "graphics/real.h"
#include "graphics/ivec.h"
#include "graphics/epsilon.h"
#include <math.h>
#include <stdio.h>
```

Classes

- struct [graphics::vec2](#)
structure for 2-dimensional real type vector and its arithmetic.
- struct [graphics::vec3](#)
structure for 3-dimensional real type vector and its arithmetic.
- struct [graphics::vec4](#)
structure for 4-dimensional real type vector and its arithmetic.

Namespaces

- [graphics](#)

Functions

- `vec2 graphics::operator+` (const vec2 &a, const vec2 &b)
- `vec2 graphics::operator+` (real a, const vec2 &b)
- `vec2 graphics::operator+` (const vec2 &a, real b)
- `vec2 graphics::operator-` (const vec2 &a, const vec2 &b)
- `vec2 graphics::operator-` (real a, const vec2 &b)
- `vec2 graphics::operator-` (const vec2 &a, real b)
- `vec2 graphics::operator*` (const vec2 &a, const vec2 &b)
- `vec2 graphics::operator*` (real a, const vec2 &b)
- `vec2 graphics::operator*` (const vec2 &a, real b)
- `vec2 graphics::operator/` (const vec2 &a, const vec2 &b)
- `vec2 graphics::operator/` (real a, const vec2 &b)
- `vec2 graphics::operator/` (const vec2 &a, real b)
- `vec2 graphics::operator+=` (vec2 &a, const vec2 &b)
- `vec2 graphics::operator+=` (vec2 &a, real b)
- `vec2 graphics::operator-=` (vec2 &a, const vec2 &b)
- `vec2 graphics::operator-=` (vec2 &a, real b)
- `vec2 graphics::operator*=` (vec2 &a, const vec2 &b)
- `vec2 graphics::operator*=` (vec2 &a, real b)
- `vec2 graphics::operator/=` (vec2 &a, const vec2 &b)
- `vec2 graphics::operator/=` (vec2 &a, real b)
- `int graphics::operator==` (const vec2 &a, const vec2 &b)
- `int graphics::operator==` (real a, const vec2 &b)
- `int graphics::operator==` (const vec2 &a, real b)
- `int graphics::operator<` (const vec2 &a, const vec2 &b)
- `int graphics::operator<` (real a, const vec2 &b)
- `int graphics::operator<` (const vec2 &a, real b)
- `int graphics::operator<=` (const vec2 &a, const vec2 &b)
- `int graphics::operator<=` (real a, const vec2 &b)
- `int graphics::operator<=` (const vec2 &a, real b)
- `int graphics::operator>` (const vec2 &a, const vec2 &b)
- `int graphics::operator>` (real a, const vec2 &b)
- `int graphics::operator>` (const vec2 &a, real b)
- `int graphics::operator>=` (const vec2 &a, const vec2 &b)
- `int graphics::operator>=` (real a, const vec2 &b)
- `int graphics::operator>=` (const vec2 &a, real b)
- `vec2 graphics::operator-` (const vec2 &a)
- `real graphics::sum` (const vec2 &a)
- `real graphics::inner` (const vec2 &a, const vec2 &b)
- `real graphics::norm` (const vec2 &a)
- `real graphics::squaredNorm` (const vec2 &a)
- `vec2 graphics::unit` (const vec2 &a)
- `real graphics::angle` (const vec2 &a, const vec2 &b)
- `vec2 graphics::proj` (const vec2 &axis, const vec2 &a)
- `vec2 graphics::absolute` (const vec2 &val)
- `void graphics::store` (FILE *fp, const vec2 &v)
- `int graphics::scan` (FILE *fp, const vec2 &v)
- `int graphics::apx_equal` (const vec2 &a, const vec2 &b)
- `int graphics::apx_equal` (const vec2 &a, const vec2 &b, real eps)
- `vec3 graphics::operator+` (const vec3 &a, const vec3 &b)
- `vec3 graphics::operator+` (real a, const vec3 &b)
- `vec3 graphics::operator+` (const vec3 &a, real b)
- `vec3 graphics::operator-` (const vec3 &a, const vec3 &b)
- `vec3 graphics::operator-` (real a, const vec3 &b)

- `vec3 graphics::operator-` (const vec3 &a, `real` b)
- `vec3 graphics::operator*` (const vec3 &a, const vec3 &b)
- `vec3 graphics::operator*` (`real` a, const vec3 &b)
- `vec3 graphics::operator*` (const vec3 &a, `real` b)
- `vec3 graphics::operator/` (const vec3 &a, const vec3 &b)
- `vec3 graphics::operator/` (`real` a, const vec3 &b)
- `vec3 graphics::operator/` (const vec3 &a, `real` b)
- `vec3 graphics::operator+=` (vec3 &a, const vec3 &b)
- `vec3 graphics::operator+=` (vec3 &a, `real` b)
- `vec3 graphics::operator-=` (vec3 &a, const vec3 &b)
- `vec3 graphics::operator-=` (vec3 &a, `real` b)
- `vec3 graphics::operator*=` (vec3 &a, const vec3 &b)
- `vec3 graphics::operator*=` (vec3 &a, `real` b)
- `vec3 graphics::operator/=` (vec3 &a, const vec3 &b)
- `vec3 graphics::operator/=` (vec3 &a, `real` b)
- `int graphics::operator==` (const vec3 &a, const vec3 &b)
- `int graphics::operator==` (`real` a, const vec3 &b)
- `int graphics::operator==` (const vec3 &a, `real` b)
- `int graphics::operator<` (const vec3 &a, const vec3 &b)
- `int graphics::operator<` (`real` a, const vec3 &b)
- `int graphics::operator<` (const vec3 &a, `real` b)
- `int graphics::operator<=` (const vec3 &a, const vec3 &b)
- `int graphics::operator<=` (`real` a, const vec3 &b)
- `int graphics::operator<=` (const vec3 &a, `real` b)
- `int graphics::operator>` (const vec3 &a, const vec3 &b)
- `int graphics::operator>` (`real` a, const vec3 &b)
- `int graphics::operator>` (const vec3 &a, `real` b)
- `int graphics::operator>=` (const vec3 &a, const vec3 &b)
- `int graphics::operator>=` (`real` a, const vec3 &b)
- `int graphics::operator>=` (const vec3 &a, `real` b)
- `vec3 graphics::operator-` (const vec3 &a)
- `vec3 graphics::absolute` (const vec3 &val)
- `real graphics::sum` (const vec3 &a)
- `real graphics::inner` (const vec3 &a, const vec3 &b)
- `real graphics::squaredNorm` (const vec3 &a)
- `real graphics::norm` (const vec3 &a)
- `vec3 graphics::unit` (const vec3 &a)
- `real graphics::angle` (const vec3 &a, const vec3 &b)
- `vec3 graphics::proj` (const vec3 &axis, const vec3 &a)
- `void graphics::store` (FILE *fp, const vec3 &v)
- `int graphics::scan` (FILE *fp, const vec3 &v)
- `int graphics::apx_equal` (const vec3 &a, const vec3 &b)
- `int graphics::apx_equal` (const vec3 &a, const vec3 &b, `real` eps)
- `vec4 graphics::operator+` (const vec4 &a, const vec4 &b)
- `vec4 graphics::operator+` (`real` a, const vec4 &b)
- `vec4 graphics::operator+` (const vec4 &a, `real` b)
- `vec4 graphics::operator-` (const vec4 &a, const vec4 &b)
- `vec4 graphics::operator-` (`real` a, const vec4 &b)
- `vec4 graphics::operator-` (const vec4 &a, `real` b)
- `vec4 graphics::operator*` (const vec4 &a, const vec4 &b)
- `vec4 graphics::operator*` (`real` a, const vec4 &b)
- `vec4 graphics::operator*` (const vec4 &a, `real` b)
- `vec4 graphics::operator/` (const vec4 &a, const vec4 &b)
- `vec4 graphics::operator/` (`real` a, const vec4 &b)
- `vec4 graphics::operator/` (const vec4 &a, `real` b)

- `vec4 graphics::operator+=` (`vec4 &a`, `const vec4 &b`)
- `vec4 graphics::operator+=` (`vec4 &a`, `real b`)
- `vec4 graphics::operator-=` (`vec4 &a`, `const vec4 &b`)
- `vec4 graphics::operator-=` (`vec4 &a`, `real b`)
- `vec4 graphics::operator*=` (`vec4 &a`, `const vec4 &b`)
- `vec4 graphics::operator*=` (`vec4 &a`, `real b`)
- `vec4 graphics::operator/=` (`vec4 &a`, `const vec4 &b`)
- `vec4 graphics::operator/=` (`vec4 &a`, `real b`)
- `int graphics::operator==` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::operator==` (`real a`, `const vec4 &b`)
- `int graphics::operator==` (`const vec4 &a`, `real b`)
- `int graphics::operator<` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::operator<` (`real a`, `const vec4 &b`)
- `int graphics::operator<` (`const vec4 &a`, `real b`)
- `int graphics::operator<=` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::operator<=` (`real a`, `const vec4 &b`)
- `int graphics::operator<=` (`const vec4 &a`, `real b`)
- `int graphics::operator>` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::operator>` (`real a`, `const vec4 &b`)
- `int graphics::operator>` (`const vec4 &a`, `real b`)
- `int graphics::operator>=` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::operator>=` (`real a`, `const vec4 &b`)
- `int graphics::operator>=` (`const vec4 &a`, `real b`)
- `vec4 graphics::operator-` (`const vec4 &a`)
- `vec4 graphics::absolute` (`const vec4 &val`)
- `real graphics::sum` (`const vec4 &a`)
- `real graphics::inner` (`const vec4 &a`, `const vec4 &b`)
- `real graphics::squaredNorm` (`const vec4 &a`)
- `real graphics::norm` (`const vec4 &a`)
- `vec4 graphics::unit` (`const vec4 &a`)
- `real graphics::angle` (`const vec4 &a`, `const vec4 &b`)
- `vec4 graphics::proj` (`const vec4 &axis`, `const vec4 &a`)
- `void graphics::store` (`FILE *fp`, `const vec4 &v`)
- `int graphics::scan` (`FILE *fp`, `const vec4 &v`)
- `int graphics::apx_equal` (`const vec4 &a`, `const vec4 &b`)
- `int graphics::apx_equal` (`const vec4 &a`, `const vec4 &b`, `real eps`)
- `vec3 graphics::cross` (`const vec3 &a`, `const vec3 &b`)

10.10 Hologram/HologramGenerator.h File Reference

```
#include <graphics/vec.h>
#include <graphics/complex.h>
#include <QtCore/QDir>
#include <QtCore/QFile>
#include <QtGui/QImage>
#include <QtWidgets/qmessagebox.h>
#include <vector>
#include <cufft.h>
#include "Hologram/fftw3.h"
```

Classes

- struct [HologramParams](#)
Structure variable for hologram paramemters.
- class [HologramGenerator](#)
Main class for generating a hologram using depth map data.

10.11 Hologram/src/HologramGenerator.cpp File Reference

```
#include "Hologram/HologramGenerator.h"  
#include <fstream>  
#include <sstream>  
#include <random>  
#include "graphics/sys.h"  
#include <QtScript/QScriptEngine>  
#include <QtCore/QRegularExpression>
```

10.12 Hologram/src/HologramGenerator_CPU.cpp File Reference

```
#include "Hologram/HologramGenerator.h"  
#include "graphics/sys.h"
```

Variables

- fftw_plan [fft_plan_fwd_](#)
- fftw_plan [fft_plan_bwd_](#)

10.12.1 Variable Documentation

10.12.1.1 [fft_plan_bwd_](#)

```
fftw_plan fft_plan_bwd_
```

Definition at line 6 of file HologramGenerator_CPU.cpp.

10.12.1.2 [fft_plan_fwd_](#)

```
fftw_plan fft_plan_fwd_
```

Definition at line 5 of file HologramGenerator_CPU.cpp.

10.13 Hologram/src/HologramGenerator_GPU.cpp File Reference

```
#include "Hologram/HologramGenerator.h"
#include "graphics/sys.h"
#include <cuda_runtime.h>
#include <cuFFT.h>
```

Macros

- #define `HANDLE_ERROR`(err) (`HandleError`(err, __FILE__, __LINE__))
- #define `HANDLE_NULL`(a)

Functions

- static void `HandleError` (cudaError_t err, const char *file, int line)
- void `cudaFFT` (CUstream_st *stream, int nx, int ny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int direction, bool bNormalized=false)
Convert data from the spatial domain to the frequency domain using 2D FFT on GPU.
- void `cudaCropFringe` (CUstream_st *stream, int nx, int ny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int cropx1, int cropx2, int cropy1, int cropy2)
Crop input data according to x, y coordinates on GPU.
- void `cudaDepthHoloKernel` (CUstream_st *stream, int pnx, int pny, cufftDoubleComplex *u_o_gpu_, unsigned char *img_src_gpu_, unsigned char *dimg_src_gpu_, double *depth_index_gpu_, int dtr, double rand_phase_val_a, double rand_phase_val_b, double carrier_phase_delay_a, double carrier_phase_delay_b, int flag_change_depth_quan, unsigned int default_depth_quan)
Find each depth plane of the input image and apply carrier phase delay to it on GPU.
- void `cudaPropagation_AngularSpKernel` (CUstream_st *stream_, int pnx, int pny, cufftDoubleComplex *input_d, cufftDoubleComplex *u_complex, double ppx, double ppy, double ssx, double ssy, double lambda, double params_k, double propagation_dist)
Angular spectrum propagation method for GPU implementation.
- void `cudaGetFringe` (CUstream_st *stream, int pnx, int pny, cufftDoubleComplex *in_field, cufftDoubleComplex *out_field, int sig_locationx, int sig_locationy, double ssx, double ssy, double ppx, double ppy, double PI)
Encode the CGH according to a signal location parameter on the GPU.
- void `cudaChangeDepthQuanKernel` (CUstream_st *stream_, int pnx, int pny, double *depth_index_gpu, unsigned char *dimg_src_gpu, int dtr, double d1, double d2, double params_num_of_depth, double params_far_depthmap, double params_near_depthmap)
Quantize depth map on the GPU, only when the number of depth quantization is not the default value (i.e. FLAG_CHANGE_DEPTH_QUANTIZATION == 1).

Variables

- cufftDoubleComplex * `u_o_gpu_`
- cufftDoubleComplex * `u_complex_gpu_`
- cufftDoubleComplex * `k_temp_d_`
- cudaStream_t `stream_`
- cudaEvent_t `start`
- cudaEvent_t `stop`

10.13.1 Macro Definition Documentation

10.13.1.1 HANDLE_ERROR

```
#define HANDLE_ERROR(  
    err ) (HandleError( err, __FILE__, __LINE__ ))
```

Definition at line 16 of file HologramGenerator_GPU.cpp.

10.13.1.2 HANDLE_NULL

```
#define HANDLE_NULL(  
    a )
```

Value:

```
{if (a == NULL) { \n                printf( "Host memory failed in %s at line %d\n", \n                        __FILE__, __LINE__ ); \n                exit( EXIT_FAILURE );}}
```

Definition at line 19 of file HologramGenerator_GPU.cpp.

10.13.2 Function Documentation

10.13.2.1 HandleError()

```
static void HandleError (  
    cudaError_t err,  
    const char * file,  
    int line ) [static]
```

Definition at line 7 of file HologramGenerator_GPU.cpp.

10.13.3 Variable Documentation

10.13.3.1 k_temp_d_

`cufftDoubleComplex* k_temp_d_`

Definition at line 26 of file HologramGenerator_GPU.cpp.

10.13.3.2 start

`cudaEvent_t start`

Definition at line 29 of file HologramGenerator_GPU.cpp.

10.13.3.3 stop

`cudaEvent_t stop`

Definition at line 29 of file HologramGenerator_GPU.cpp.

10.13.3.4 stream_

`cudaStream_t stream_`

Definition at line 28 of file HologramGenerator_GPU.cpp.

10.13.3.5 u_complex_gpu_

`cufftDoubleComplex* u_complex_gpu_`

Definition at line 25 of file HologramGenerator_GPU.cpp.

10.13.3.6 u_o_gpu_

`cufftDoubleComplex* u_o_gpu_`

Definition at line 24 of file HologramGenerator_GPU.cpp.

10.14 Hologram/src/HologramKernel.cu File Reference

10.15 HologramDepthmap/hologramdepthmap.cpp File Reference

```
#include "hologramdepthmap.h"  
#include <QtCore/QTime>
```

10.16 HologramDepthmap/hologramdepthmap.h File Reference

```
#include <QtWidgets/QMainWindow>  
#include "ui_hologramdepthmap.h"  
#include "Hologram/HologramGenerator.h"  
#include "graphics/sys.h"
```

Classes

- class [HologramDepthmap](#)

Test class for executing the sample program, which shows how to use a hologram library.

10.17 HologramDepthmap/main.cpp File Reference

```
#include "hologramdepthmap.h"  
#include <QtWidgets/QApplication>
```

Functions

- int [main](#) (int argc, char *argv[])

10.17.1 Function Documentation

10.17.1.1 main()

```
int main (  
    int argc,  
    char * argv[ ] )
```

Definition at line 4 of file main.cpp.

Index

- `_MAXDOUBLE`
 - `real.h`, 147
- `_MAXFLOAT`
 - `real.h`, 147
- `_MINDOUBLE`
 - `real.h`, 147
- `_MINFLOAT`
 - `real.h`, 147
- `~HologramDepthmap`
 - `HologramDepthmap`, 101
- `~HologramGenerator`
 - `HologramGenerator`, 107
- a
 - Complex, 100
- absolute
 - graphics, 42
- `alpha_map_`
 - `HologramGenerator`, 110
- angle
 - graphics, 43
- `angle_tolerance`
 - graphics, 91
- `apx_equal`
 - graphics, 43–45
- arg
 - Complex, 96
- b
 - Complex, 100
- `Calc_Holo_CPU`
 - Generation Hologram, 21
- `Calc_Holo_GPU`
 - Generation Hologram, 22
- `Calc_Holo_by_Depth`
 - Generation Hologram, 21
- `change_depth_quan_CPU`
 - Computing Depth Value, 19
- `change_depth_quan_GPU`
 - Computing Depth Value, 19
- `circshift`
 - Reconstruction, 29
- Complex, 95
 - a, 100
 - arg, 96
 - b, 100
 - Complex, 96
 - conj, 96
 - euler, 97
 - mag, 97
 - mag2, 97
 - operator<<, 99
 - operator*, 98, 99
 - operator*==, 97
 - operator+, 99
 - operator+=, 97
 - operator-, 99
 - operator-=, 98
 - operator/, 99
 - operator/==, 98
 - operator==, 98
- `complex.h`
 - PI, 143
 - TWO_PI, 143
- Computing Depth Value, 19
 - `change_depth_quan_CPU`, 19
 - `change_depth_quan_GPU`, 19
 - `GetDepthValues`, 19
- conj
 - Complex, 96
- cross
 - graphics, 45
- `cudaChangeDepthQuanKernel`
 - GPU Modules, 31
- `cudaCropFringe`
 - GPU Modules, 32
- `cudaDepthHoloKernel`
 - GPU Modules, 33
- `cudaFFT`
 - GPU Modules, 33
- `cudaGetFringe`
 - GPU Modules, 34
- `cudaPropagation_AngularSpKernel`
 - GPU Modules, 35
- DEFAULT_DEPTH_QUANTIZATION
 - `HologramGenerator`, 110
- DEPTH_PREFIX
 - `HologramGenerator`, 110
- `depth_index_`
 - `HologramGenerator`, 110
- `depth_index_gpu_`
 - `HologramGenerator`, 110
- `dimg_src_gpu_`
 - `HologramGenerator`, 110
- `dlevel_`
 - `HologramGenerator`, 111
- `dlevel_transform_`
 - `HologramGenerator`, 111

- dmap_
 - HologramGenerator, 111
- dmap_src_
 - HologramGenerator, 111
- dstep_
 - HologramGenerator, 111
- Encoding, 26
 - encoding_CPU, 26
 - encoding_GPU, 26
 - Encoding_Symmetrization, 27
- encoding_CPU
 - Encoding, 26
- encoding_GPU
 - Encoding, 26
- Encoding_Method_
 - HologramGenerator, 111
- Encoding_Symmetrization
 - Encoding, 27
- epsilon
 - graphics, 92
- euler
 - Complex, 97
- exponent_complex
 - HologramGenerator, 107
- eye_center_xy_
 - HologramGenerator, 112
- eye_length_
 - HologramGenerator, 112
- eye_pupil_diameter_
 - HologramGenerator, 112
- f_field_
 - HologramGenerator, 112
- FLAG_CHANGE_DEPTH_QUANTIZATION
 - HologramGenerator, 112
- FLAG_STATIC_IMAGE
 - HologramGenerator, 112
- FLOG
 - sys.h, 153
- far_depthmap
 - HologramParams, 118
- fft_plan_bwd_
 - HologramGenerator_CPU.cpp, 158
- fft_plan_fwd_
 - HologramGenerator_CPU.cpp, 158
- fftShift
 - HologramGenerator, 107
- fftwShift
 - HologramGenerator, 108
- field_lens
 - HologramParams, 118
- file_log
 - sys.cpp, 150
- file_read_open
 - sys.cpp, 150
 - sys.h, 153
- file_write_open
 - sys.cpp, 150
- sys.h, 153
- focus_distance_
 - HologramGenerator, 113
- fp
 - sys.cpp, 151
- GPU Modules, 31
 - cudaChangeDepthQuanKernel, 31
 - cudaCropFringe, 32
 - cudaDepthHoloKernel, 33
 - cudaFFT, 33
 - cudaGetFringe, 34
 - cudaPropagation_AngularSpKernel, 35
- GenHologram
 - HologramDepthmap, 101
- GenerateHologram
 - Generation Hologram, 22
- Generation Hologram, 21
 - Calc_Holo_CPU, 21
 - Calc_Holo_GPU, 22
 - Calc_Holo_by_Depth, 21
 - GenerateHologram, 22
 - Propagation_AngularSpectrum_CPU, 23
 - Propagation_AngularSpectrum_GPU, 23
- get_rand_phase_value
 - HologramGenerator, 108
- get_shift_phase_value
 - HologramGenerator, 109
- GetDepthValues
 - Computing Depth Value, 19
- graphics, 37
 - absolute, 42
 - angle, 43
 - angle_tolerance, 91
 - apx_equal, 43–45
 - cross, 45
 - epsilon, 92
 - inner, 45
 - intersection_epsilon, 92
 - norm, 46
 - operator!=, 46, 47
 - operator<, 69–72
 - operator<=, 73–76
 - operator>, 80–83
 - operator>=, 83–87
 - operator*, 47–50
 - operator*==, 50–53
 - operator+, 53–56
 - operator+==, 56–59
 - operator-, 59–63
 - operator-=, 64–66
 - operator/, 66–68
 - operator/==, 68, 69
 - operator==, 76–80
 - proj, 87
 - reset_u_epsilon, 88
 - reset_zero_epsilon, 88
 - save_zero_epsilon, 92
 - scan, 88

- set_u_epsilon, 89
- set_zero_epsilon, 89
- sqrt_epsilon, 92
- squaredNorm, 89
- store, 90
- sum, 90, 91
- unit, 91
- unset_value, 92
- user_epsilon, 92
- zero_epsilon, 93
- zero_tolerance, 93
- graphics/complex.h, 143
- graphics/epsilon.h, 144
- graphics/ivec.h, 144
- graphics/real.h, 146
- graphics/src/epsilon.cpp, 149
- graphics/src/sys.cpp, 149
- graphics/src/vec.cpp, 152
- graphics/sys.h, 152
- graphics/vec.h, 154
- graphics::ivec2, 120
 - ivec2, 121
 - n, 123
 - operator(), 122
 - operator=, 122
 - operator[], 122
 - v, 123
- graphics::ivec3, 123
 - ivec3, 124
 - n, 126
 - operator(), 125
 - operator=, 125
 - operator[], 125
 - v, 126
- graphics::ivec4, 126
 - ivec4, 127
 - n, 129
 - operator(), 128
 - operator=, 128
 - operator[], 128
 - v, 129
- graphics::vec2, 129
 - is_parallel, 131
 - is_perpendicular, 131
 - is_tiny, 131
 - is_zero, 131
 - length, 132
 - n, 133
 - operator(), 132
 - operator=, 132
 - operator[], 132
 - perpendicular, 133
 - unit, 133
 - v, 133
 - vec2, 130, 131
- graphics::vec3, 134
 - is_parallel, 136
 - is_perpendicular, 136
 - is_tiny, 136
 - is_zero, 136
 - length, 136
 - n, 138
 - operator(), 137
 - operator=, 137
 - operator[], 137
 - perpendicular, 137, 138
 - unit, 138
 - v, 138
 - vec3, 135
- graphics::vec4, 139
 - is_tiny, 140
 - is_zero, 140
 - length, 141
 - n, 142
 - operator(), 141
 - operator=, 141
 - operator[], 141
 - unit, 142
 - v, 142
 - vec4, 139, 140
- HANDLE_ERROR
 - HologramGenerator_GPU.cpp, 160
- HANDLE_NULL
 - HologramGenerator_GPU.cpp, 160
- HandleError
 - HologramGenerator_GPU.cpp, 160
- hh_complex_
 - HologramGenerator, 113
- Hologram/HologramGenerator.h, 157
- Hologram/src/HologramGenerator.cpp, 158
- Hologram/src/HologramGenerator_CPU.cpp, 158
- Hologram/src/HologramGenerator_GPU.cpp, 159
- Hologram/src/HologramKernel.cu, 162
- hologram_
 - HologramDepthmap, 102
- HologramDepthmap, 100
 - ~HologramDepthmap, 101
 - GenHologram, 101
 - hologram_, 102
 - HologramDepthmap, 101
 - ReconImage, 101
 - ui, 102
- HologramDepthmap/hologramdepthmap.cpp, 162
- HologramDepthmap/hologramdepthmap.h, 162
- HologramDepthmap/main.cpp, 162
- HologramGenerator, 102
 - ~HologramGenerator, 107
 - alpha_map_, 110
 - DEFAULT_DEPTH_QUANTIZATION, 110
 - DEPTH_PREFIX, 110
 - depth_index_, 110
 - depth_index_gpu_, 110
 - dimg_src_gpu_, 110
 - dlevel_, 111
 - dlevel_transform_, 111
 - dmap_, 111

- dmap_src_, 111
- dstep_, 111
- Encoding_Method_, 111
- exponent_complex, 107
- eye_center_xy_, 112
- eye_length_, 112
- eye_pupil_diameter_, 112
- f_field_, 112
- FLAG_CHANGE_DEPTH_QUANTIZATION, 112
- FLAG_STATIC_IMAGE, 112
- fftShift, 107
- fftwShift, 108
- focus_distance_, 113
- get_rand_phase_value, 108
- get_shift_phase_value, 109
- hh_complex_, 113
- HologramGenerator, 107
- IMAGE_PREFIX, 113
- img_src_, 113
- img_src_gpu_, 113
- isCPU_, 113
- NUMBER_OF_DEPTH_QUANTIZATION, 114
- NUMBER_OF_DIGIT_OF_FRAME_NUMBERING, 114
- NUMBER_OF_FRAME, 114
- params_, 114
- Pixel_pitch_xy_, 114
- Propagation_Method_, 114
- RANDOM_PHASE, 115
- RESULT_FOLDER, 115
- RESULT_PREFIX, 115
- SLM_pixel_number_xy_, 116
- SOURCE_FOLDER, 116
- START_OF_FRAME_NUMBERING, 116
- setMode, 109
- sim_final_, 115
- sim_from_, 115
- sim_step_num_, 115
- sim_to_, 116
- sim_type_, 116
- Simulation_Result_File_Prefix_, 116
- test_pixel_number_scale_, 117
- Transform_Method_, 117
- u255_fringe_, 117
- U_complex_, 117
- WAVELENGTH, 117
- HologramGenerator_CPU.cpp
 - fft_plan_bwd_, 158
 - fft_plan_fwd_, 158
- HologramGenerator_GPU.cpp
 - HANDLE_ERROR, 160
 - HANDLE_NULL, 160
 - HandleError, 160
 - k_temp_d_, 160
 - start, 161
 - stop, 161
 - stream_, 161
 - u_complex_gpu_, 161
 - u_o_gpu_, 161
- HologramParams, 118
 - far_depthmap, 118
 - field_lens, 118
 - k, 119
 - lambda, 119
 - near_depthmap, 119
 - num_of_depth, 119
 - pn, 119
 - pp, 120
 - render_depth, 120
 - ss, 120
- IMAGE_PREFIX
 - HologramGenerator, 113
- img_src_
 - HologramGenerator, 113
- img_src_gpu_
 - HologramGenerator, 113
- init_CPU
 - Initialize, 15
- init_GPU
 - Initialize, 15
- Initialize, 15
 - init_CPU, 15
 - init_GPU, 15
 - initialize, 16
 - readConfig, 16
- initialize
 - Initialize, 16
- inner
 - graphics, 45
- intersection_epsilon
 - graphics, 92
- is_parallel
 - graphics::vec2, 131
 - graphics::vec3, 136
- is_perpendicular
 - graphics::vec2, 131
 - graphics::vec3, 136
- is_tiny
 - graphics::vec2, 131
 - graphics::vec3, 136
 - graphics::vec4, 140
- is_zero
 - graphics::vec2, 131
 - graphics::vec3, 136
 - graphics::vec4, 140
- isCPU_
 - HologramGenerator, 113
- ivec2
 - graphics::ivec2, 121
- ivec3
 - graphics::ivec3, 124
- ivec4
 - graphics::ivec4, 127
- k
 - HologramParams, 119

- k_temp_d_
 - HologramGenerator_GPU.cpp, 160
- lambda
 - HologramParams, 119
- length
 - graphics::vec2, 132
 - graphics::vec3, 136
 - graphics::vec4, 141
- Loading Data, 17
 - prepare_inputdata_CPU, 17
 - prepare_inputdata_GPU, 17
 - ReadImageDepth, 18
- M_PI
 - real.h, 148
- MAXREAL
 - real.h, 148
- MINREAL
 - real.h, 148
- mag
 - Complex, 97
- mag2
 - Complex, 97
- main
 - main.cpp, 162
- main.cpp
 - main, 162
- n
 - graphics::ivec2, 123
 - graphics::ivec3, 126
 - graphics::ivec4, 129
 - graphics::vec2, 133
 - graphics::vec3, 138
 - graphics::vec4, 142
- NUMBER_OF_DEPTH_QUANTIZATION
 - HologramGenerator, 114
- NUMBER_OF_DIGIT_OF_FRAME_NUMBERING
 - HologramGenerator, 114
- NUMBER_OF_FRAME
 - HologramGenerator, 114
- near_depthmap
 - HologramParams, 119
- norm
 - graphics, 46
- num_of_depth
 - HologramParams, 119
- operator!=
 - graphics, 46, 47
- operator<
 - graphics, 69–72
- operator<<
 - Complex, 99
- operator<=
 - graphics, 73–76
- operator>
 - graphics, 80–83
- operator>=
 - graphics, 83–87
- operator*
 - Complex, 98, 99
 - graphics, 47–50
- operator*=
 - Complex, 97
 - graphics, 50–53
- operator()
 - graphics::ivec2, 122
 - graphics::ivec3, 125
 - graphics::ivec4, 128
 - graphics::vec2, 132
 - graphics::vec3, 137
 - graphics::vec4, 141
- operator+
 - Complex, 99
 - graphics, 53–56
- operator+=
 - Complex, 97
 - graphics, 56–59
- operator-
 - Complex, 99
 - graphics, 59–63
- operator-=
 - Complex, 98
 - graphics, 64–66
- operator/
 - Complex, 99
 - graphics, 66–68
- operator/=
 - Complex, 98
 - graphics, 68, 69
- operator=
 - Complex, 98
 - graphics::ivec2, 122
 - graphics::ivec3, 125
 - graphics::ivec4, 128
 - graphics::vec2, 132
 - graphics::vec3, 137
 - graphics::vec4, 141
- operator==
 - graphics, 76–80
- operator[]
 - graphics::ivec2, 122
 - graphics::ivec3, 125
 - graphics::ivec4, 128
 - graphics::vec2, 132
 - graphics::vec3, 137
 - graphics::vec4, 141
- params_
 - HologramGenerator, 114
- perpendicular
 - graphics::vec2, 133
 - graphics::vec3, 137, 138
- PI
 - complex.h, 143
- Pixel_pitch_xy_

- HologramGenerator, 114
- pn
 - HologramParams, 119
- pp
 - HologramParams, 120
- prepare_inputdata_CPU
 - Loading Data, 17
- prepare_inputdata_GPU
 - Loading Data, 17
- proj
 - graphics, 87
- Propagation_AngularSpectrum_CPU
 - Generation Hologram, 23
- Propagation_AngularSpectrum_GPU
 - Generation Hologram, 23
- Propagation_Method_
 - HologramGenerator, 114
- RANDOM_PHASE
 - HologramGenerator, 115
- REAL_T_IS_FLOAT
 - real.h, 148
- RESULT_FOLDER
 - HologramGenerator, 115
- RESULT_PREFIX
 - HologramGenerator, 115
- readConfig
 - Initialize, 16
- ReadImageDepth
 - Loading Data, 18
- real
 - real.h, 148
- real.h
 - _MAXDOUBLE, 147
 - _MAXFLOAT, 147
 - _MINDOUBLE, 147
 - _MINFLOAT, 147
 - M_PI, 148
 - MAXREAL, 148
 - MINREAL, 148
 - REAL_T_IS_FLOAT, 148
 - real, 148
 - real_t, 148
- real_t
 - real.h, 148
- ReconImage
 - HologramDepthmap, 101
- ReconstructImage
 - Reconstruction, 29
- Reconstruction, 29
 - circshift, 29
 - ReconstructImage, 29
 - Reconstruction, 29
 - Test_Propagation_to_Eye_Pupil, 30
 - Write_Simulation_image, 30
- render_depth
 - HologramParams, 120
- reset_u_epsilon
 - graphics, 88
- reset_zero_epsilon
 - graphics, 88
- SLM_pixel_number_xy_
 - HologramGenerator, 116
- SOURCE_FOLDER
 - HologramGenerator, 116
- START_OF_FRAME_NUMBERING
 - HologramGenerator, 116
- save_zero_epsilon
 - graphics, 92
- scan
 - graphics, 88
- set_u_epsilon
 - graphics, 89
- set_zero_epsilon
 - graphics, 89
- setMode
 - HologramGenerator, 109
- sim_final_
 - HologramGenerator, 115
- sim_from_
 - HologramGenerator, 115
- sim_step_num_
 - HologramGenerator, 115
- sim_to_
 - HologramGenerator, 116
- sim_type_
 - HologramGenerator, 116
- Simulation_Result_File_Prefix_
 - HologramGenerator, 116
- sqrt_epsilon
 - graphics, 92
- squaredNorm
 - graphics, 89
- ss
 - HologramParams, 120
- start
 - HologramGenerator_GPU.cpp, 161
- stop
 - HologramGenerator_GPU.cpp, 161
- store
 - graphics, 90
- stream_
 - HologramGenerator_GPU.cpp, 161
- string_cat
 - sys.cpp, 151
 - sys.h, 153
- string_cmp
 - sys.cpp, 151
 - sys.h, 154
- string_cpy
 - sys.cpp, 151
 - sys.h, 154
- sum
 - graphics, 90, 91
- sys.cpp
 - file_log, 150
 - file_read_open, 150

- file_write_open, [150](#)
- fp, [151](#)
- string_cat, [151](#)
- string_cmp, [151](#)
- string_cpy, [151](#)
- sys.h
 - FLOG, [153](#)
 - file_read_open, [153](#)
 - file_write_open, [153](#)
 - string_cat, [153](#)
 - string_cmp, [154](#)
 - string_cpy, [154](#)
- TWO_PI
 - complex.h, [143](#)
- Test_Propagation_to_Eye_Pupil
 - Reconstruction, [30](#)
- test_pixel_number_scale_
 - HologramGenerator, [117](#)
- Transform, [20](#)
 - TransformViewingWindow, [20](#)
- Transform_Method_
 - HologramGenerator, [117](#)
- TransformViewingWindow
 - Transform, [20](#)
- u255_fringe_
 - HologramGenerator, [117](#)
- U_complex_
 - HologramGenerator, [117](#)
- u_complex_gpu_
 - HologramGenerator_GPU.cpp, [161](#)
- u_o_gpu_
 - HologramGenerator_GPU.cpp, [161](#)
- ui
 - HologramDepthmap, [102](#)
- unit
 - graphics, [91](#)
 - graphics::vec2, [133](#)
 - graphics::vec3, [138](#)
 - graphics::vec4, [142](#)
- unset_value
 - graphics, [92](#)
- user_epsilon
 - graphics, [92](#)
- v
 - graphics::ivec2, [123](#)
 - graphics::ivec3, [126](#)
 - graphics::ivec4, [129](#)
 - graphics::vec2, [133](#)
 - graphics::vec3, [138](#)
 - graphics::vec4, [142](#)
- vec2
 - graphics::vec2, [130](#), [131](#)
- vec3
 - graphics::vec3, [135](#)
- vec4
 - graphics::vec4, [139](#), [140](#)
- WAVELENGTH
 - HologramGenerator, [117](#)
- Write_Result_image
 - Writing Image, [28](#)
- Write_Simulation_image
 - Reconstruction, [30](#)
- Writing Image, [28](#)
 - Write_Result_image, [28](#)
- zero_epsilon
 - graphics, [93](#)
- zero_tolerance
 - graphics, [93](#)