

Road To Zero

Generated by Doxygen 1.9.1

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 AssetsManager Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 AssetsManager()	8
4.1.2.2 ~AssetsManager()	9
4.1.3 Member Function Documentation	9
4.1.3.1 __loadSoundBuffer()	9
4.1.3.2 clear()	10
4.1.3.3 getCurrentTrackKey()	11
4.1.3.4 getFont()	11
4.1.3.5 getSound()	12
4.1.3.6 getSoundBuffer()	12
4.1.3.7 getTexture()	13
4.1.3.8 getTrackStatus()	13
4.1.3.9 loadFont()	14
4.1.3.10 loadSound()	14
4.1.3.11 loadTexture()	15
4.1.3.12 loadTrack()	16
4.1.3.13 nextTrack()	17
4.1.3.14 pauseTrack()	17
4.1.3.15 playTrack()	17
4.1.3.16 previousTrack()	17
4.1.3.17 stopTrack()	18
4.1.4 Member Data Documentation	18
4.1.4.1 current_track	18
4.1.4.2 font_map	18
4.1.4.3 sound_map	18
4.1.4.4 soundbuffer_map	18
4.1.4.5 texture_map	19
4.1.4.6 track_map	19
4.2 ContextMenu Class Reference	19
4.2.1 Detailed Description	21
4.2.2 Constructor & Destructor Documentation	21

4.2.2.1 ContextMenu()	21
4.2.2.2 ~ContextMenu()	22
4.2.3 Member Function Documentation	22
4.2.3.1 __drawConsoleScreenFrame()	22
4.2.3.2 __drawConsoleText()	23
4.2.3.3 __drawVisualScreenFrame()	24
4.2.3.4 __handleKeyPressEvents()	24
4.2.3.5 __handleMouseButtonEvents()	25
4.2.3.6 __sendQuitGameMessage()	25
4.2.3.7 __sendRestartGameMessage()	26
4.2.3.8 __setConsoleState()	26
4.2.3.9 __setConsoleString()	26
4.2.3.10 __setUpConsoleScreen()	27
4.2.3.11 __setUpConsoleScreenFrame()	27
4.2.3.12 __setUpMenuFrame()	29
4.2.3.13 __setUpVisualScreen()	30
4.2.3.14 __setUpVisualScreenFrame()	30
4.2.3.15 draw()	32
4.2.3.16 processEvent()	32
4.2.3.17 processMessage()	32
4.2.4 Member Data Documentation	33
4.2.4.1 assets_manager_ptr	33
4.2.4.2 console_screen	33
4.2.4.3 console_screen_frame_bottom	34
4.2.4.4 console_screen_frame_left	34
4.2.4.5 console_screen_frame_right	34
4.2.4.6 console_screen_frame_top	34
4.2.4.7 console_state	34
4.2.4.8 console_string	34
4.2.4.9 console_string_changed	35
4.2.4.10 console_substring_idx	35
4.2.4.11 event_ptr	35
4.2.4.12 frame	35
4.2.4.13 game_menu_up	35
4.2.4.14 menu_frame	35
4.2.4.15 message_hub_ptr	36
4.2.4.16 position_x	36
4.2.4.17 position_y	36
4.2.4.18 render_window_ptr	36
4.2.4.19 visual_screen	36
4.2.4.20 visual_screen_frame_bottom	36
4.2.4.21 visual_screen_frame_left	37

4.2.4.22 visual_screen_frame_right	37
4.2.4.23 visual_screen_frame_top	37
4.3 DieselGenerator Class Reference	37
4.3.1 Detailed Description	39
4.3.2 Constructor & Destructor Documentation	39
4.3.2.1 DieselGenerator()	39
4.3.2.2 ~DieselGenerator()	40
4.3.3 Member Function Documentation	40
4.3.3.1 __handleKeyPressEvents()	40
4.3.3.2 __handleMouseButtonEvents()	41
4.3.3.3 __setUpTileImprovementSpriteAnimated()	41
4.3.3.4 draw()	42
4.3.3.5 processEvent()	42
4.3.3.6 processMessage()	43
4.3.4 Member Data Documentation	43
4.3.4.1 skip_smoke_processing	43
4.3.4.2 smoke_da	43
4.3.4.3 smoke_dx	43
4.3.4.4 smoke_dy	43
4.3.4.5 smoke_prob	44
4.3.4.6 smoke_sprite_list	44
4.4 EnergyStorageSystem Class Reference	44
4.4.1 Detailed Description	45
4.4.2 Constructor & Destructor Documentation	46
4.4.2.1 EnergyStorageSystem()	46
4.4.2.2 ~EnergyStorageSystem()	46
4.4.3 Member Function Documentation	47
4.4.3.1 __handleKeyPressEvents()	47
4.4.3.2 __handleMouseButtonEvents()	47
4.4.3.3 __setUpTileImprovementSpriteStatic()	48
4.4.3.4 draw()	48
4.4.3.5 processEvent()	48
4.4.3.6 processMessage()	49
4.5 Game Class Reference	49
4.5.1 Detailed Description	51
4.5.2 Constructor & Destructor Documentation	51
4.5.2.1 Game()	51
4.5.2.2 ~Game()	52
4.5.3 Member Function Documentation	52
4.5.3.1 __draw()	52
4.5.3.2 __drawFrameClockOverlay()	53
4.5.3.3 __drawHUD()	53

4.5.3.4	__handleKeyPressEvents()	55
4.5.3.5	__handleMouseButtonEvents()	55
4.5.3.6	__insufficientCreditsAlarm()	56
4.5.3.7	__processEvent()	57
4.5.3.8	__processMessage()	57
4.5.3.9	__sendGameStateMessage()	58
4.5.3.10	__toggleFrameClockOverlay()	59
4.5.3.11	run()	60
4.5.4	Member Data Documentation	60
4.5.4.1	assets_manager_ptr	60
4.5.4.2	clock	61
4.5.4.3	context_menu_ptr	61
4.5.4.4	credits	61
4.5.4.5	cumulative_emissions_tonnes	61
4.5.4.6	demand_MWh	61
4.5.4.7	event	61
4.5.4.8	frame	62
4.5.4.9	game_loop_broken	62
4.5.4.10	game_phase	62
4.5.4.11	hex_map_ptr	62
4.5.4.12	message_hub	62
4.5.4.13	month	62
4.5.4.14	population	63
4.5.4.15	quit_game	63
4.5.4.16	render_window_ptr	63
4.5.4.17	show_frame_clock_overlay	63
4.5.4.18	time_since_start_s	63
4.5.4.19	turn	63
4.5.4.20	year	64
4.6	HexMap Class Reference	64
4.6.1	Detailed Description	66
4.6.2	Constructor & Destructor Documentation	66
4.6.2.1	HexMap()	66
4.6.2.2	~HexMap()	67
4.6.3	Member Function Documentation	67
4.6.3.1	__assembleHexMap()	68
4.6.3.2	__assessNeighbours()	68
4.6.3.3	__buildDrawOrderVector()	68
4.6.3.4	__enforceOceanContinuity()	69
4.6.3.5	__getMajorityTileType()	70
4.6.3.6	__getNeighboursVector()	71
4.6.3.7	__getNoise()	71

4.6.3.8	__getSelectedTile()	73
4.6.3.9	__getValidMapIndexPositions()	73
4.6.3.10	__handleKeyPressEvents()	74
4.6.3.11	__handleMouseButtonEvents()	75
4.6.3.12	__isLakeTouchingOcean()	75
4.6.3.13	__layTiles()	76
4.6.3.14	__procedurallyGenerateTileResources()	78
4.6.3.15	__procedurallyGenerateTileTypes()	78
4.6.3.16	__sendNoTileSelectedMessage()	79
4.6.3.17	__setUpGlassScreen()	79
4.6.3.18	__smoothTileTypes()	80
4.6.3.19	assess()	80
4.6.3.20	clear()	80
4.6.3.21	draw()	81
4.6.3.22	processEvent()	82
4.6.3.23	processMessage()	82
4.6.3.24	reroll()	83
4.6.3.25	toggleResourceOverlay()	83
4.6.4	Member Data Documentation	83
4.6.4.1	assets_manager_ptr	83
4.6.4.2	border_tiles_vec	84
4.6.4.3	event_ptr	84
4.6.4.4	frame	84
4.6.4.5	glass_screen	84
4.6.4.6	hex_draw_order_vec	84
4.6.4.7	hex_map	84
4.6.4.8	message_hub_ptr	85
4.6.4.9	n_layers	85
4.6.4.10	n_tiles	85
4.6.4.11	position_x	85
4.6.4.12	position_y	85
4.6.4.13	render_window_ptr	85
4.6.4.14	show_resource	86
4.6.4.15	tile_position_x_vec	86
4.6.4.16	tile_position_y_vec	86
4.6.4.17	tile_selected	86
4.7	HexTile Class Reference	86
4.7.1	Detailed Description	90
4.7.2	Constructor & Destructor Documentation	90
4.7.2.1	HexTile()	91
4.7.2.2	~HexTile()	92
4.7.3	Member Function Documentation	92

4.7.3.1 __buildDieselGenerator()	92
4.7.3.2 __buildEnergyStorage()	93
4.7.3.3 __buildSettlement()	93
4.7.3.4 __buildSolarPV()	94
4.7.3.5 __buildTidalTurbine()	94
4.7.3.6 __buildWaveEnergyConverter()	95
4.7.3.7 __buildWindTurbine()	95
4.7.3.8 __clearDecoration()	96
4.7.3.9 __closeBuildMenu()	97
4.7.3.10 __getTileCoordsSubstring()	97
4.7.3.11 __getTileImprovementSubstring()	97
4.7.3.12 __getTileOptionsSubstring()	98
4.7.3.13 __getTileResourceSubstring()	99
4.7.3.14 __getTileTypeSubstring()	100
4.7.3.15 __handleKeyPressEvents()	101
4.7.3.16 __handleMouseButtonEvents()	104
4.7.3.17 __isClicked()	105
4.7.3.18 __openBuildMenu()	105
4.7.3.19 __sendAssessNeighboursMessage()	106
4.7.3.20 __sendCreditsSpentMessage()	106
4.7.3.21 __sendGameStateRequest()	106
4.7.3.22 __sendInsufficientCreditsMessage()	107
4.7.3.23 __sendTileSelectedMessage()	107
4.7.3.24 __sendTileStateMessage()	107
4.7.3.25 __sendUpdateGamePhaseMessage()	108
4.7.3.26 __setIsSelected()	108
4.7.3.27 __setResourceText()	109
4.7.3.28 __setUpBuildMenu()	110
4.7.3.29 __setUpBuildOption()	111
4.7.3.30 __setUpDieselGeneratorBuildOption()	112
4.7.3.31 __setUpEnergyStorageSystemBuildOption()	112
4.7.3.32 __setUpMagnifyingGlassSprite()	113
4.7.3.33 __setUpNodeSprite()	113
4.7.3.34 __setUpResourceChipSprite()	113
4.7.3.35 __setUpSelectOutlineSprite()	114
4.7.3.36 __setUpSolarPVBuildOption()	114
4.7.3.37 __setUpTidalTurbineBuildOption()	115
4.7.3.38 __setUpTileExplosionReel()	115
4.7.3.39 __setUpTileSprite()	116
4.7.3.40 __setUpWaveEnergyConverterBuildOption()	116
4.7.3.41 __setUpWindTurbineBuildOption()	116
4.7.3.42 assess()	117

4.7.3.43 decorateTile()	117
4.7.3.44 draw()	119
4.7.3.45 processEvent()	120
4.7.3.46 processMessage()	120
4.7.3.47 setTileResource() [1/2]	121
4.7.3.48 setTileResource() [2/2]	121
4.7.3.49 setTileType() [1/2]	122
4.7.3.50 setTileType() [2/2]	122
4.7.3.51 toggleResourceOverlay()	123
4.7.4 Member Data Documentation	123
4.7.4.1 assets_manager_ptr	124
4.7.4.2 build_menu_backing	124
4.7.4.3 build_menu_backing_text	124
4.7.4.4 build_menu_open	124
4.7.4.5 build_menu_options_text_vec	124
4.7.4.6 build_menu_options_vec	124
4.7.4.7 credits	125
4.7.4.8 decoration_cleared	125
4.7.4.9 draw_explosion	125
4.7.4.10 event_ptr	125
4.7.4.11 explosion_frame	125
4.7.4.12 explosion_sprite_reel	125
4.7.4.13 frame	126
4.7.4.14 game_phase	126
4.7.4.15 has_improvement	126
4.7.4.16 is_selected	126
4.7.4.17 magnifying_glass_sprite	126
4.7.4.18 major_radius	126
4.7.4.19 message_hub_ptr	127
4.7.4.20 minor_radius	127
4.7.4.21 node_sprite	127
4.7.4.22 position_x	127
4.7.4.23 position_y	127
4.7.4.24 render_window_ptr	127
4.7.4.25 resource_assessed	128
4.7.4.26 resource_assessment	128
4.7.4.27 resource_chip_sprite	128
4.7.4.28 resource_text	128
4.7.4.29 select_outline_sprite	128
4.7.4.30 show_node	128
4.7.4.31 show_resource	129
4.7.4.32 tile_decoration_sprite	129

4.7.4.33 tile_improvement_ptr	129
4.7.4.34 tile_resource	129
4.7.4.35 tile_sprite	129
4.7.4.36 tile_type	129
4.8 Message Struct Reference	130
4.8.1 Detailed Description	130
4.8.2 Member Data Documentation	130
4.8.2.1 bool_payload	130
4.8.2.2 channel	130
4.8.2.3 double_payload	131
4.8.2.4 int_payload	131
4.8.2.5 string_payload	131
4.8.2.6 subject	131
4.9 MessageHub Class Reference	131
4.9.1 Detailed Description	132
4.9.2 Constructor & Destructor Documentation	132
4.9.2.1 MessageHub()	132
4.9.2.2 ~MessageHub()	133
4.9.3 Member Function Documentation	133
4.9.3.1 addChannel()	133
4.9.3.2 clear()	133
4.9.3.3 clearMessages()	134
4.9.3.4 hasTraffic()	134
4.9.3.5 isEmpty()	134
4.9.3.6 popMessage()	135
4.9.3.7 receiveMessage()	136
4.9.3.8 removeChannel()	136
4.9.3.9 sendMessage()	137
4.9.4 Member Data Documentation	137
4.9.4.1 message_map	137
4.10 Settlement Class Reference	138
4.10.1 Detailed Description	139
4.10.2 Constructor & Destructor Documentation	139
4.10.2.1 Settlement()	140
4.10.2.2 ~Settlement()	140
4.10.3 Member Function Documentation	141
4.10.3.1 __handleKeyPressEvents()	141
4.10.3.2 __handleMouseButtonEvents()	141
4.10.3.3 __setUpTileImprovementSpriteStatic()	142
4.10.3.4 draw()	142
4.10.3.5 processEvent()	143
4.10.3.6 processMessage()	143

4.10.4 Member Data Documentation	144
4.10.4.1 skip_smoke_processing	144
4.10.4.2 smoke_da	144
4.10.4.3 smoke_dx	144
4.10.4.4 smoke_dy	144
4.10.4.5 smoke_prob	144
4.10.4.6 smoke_sprite_list	145
4.11 SolarPV Class Reference	145
4.11.1 Detailed Description	146
4.11.2 Constructor & Destructor Documentation	146
4.11.2.1 SolarPV()	146
4.11.2.2 ~SolarPV()	147
4.11.3 Member Function Documentation	147
4.11.3.1 __handleKeyPressEvents()	147
4.11.3.2 __handleMouseButtonEvents()	148
4.11.3.3 __setUpTileImprovementSpriteStatic()	148
4.11.3.4 draw()	149
4.11.3.5 processEvent()	149
4.11.3.6 processMessage()	150
4.12 TidalTurbine Class Reference	150
4.12.1 Detailed Description	151
4.12.2 Constructor & Destructor Documentation	152
4.12.2.1 TidalTurbine()	152
4.12.2.2 ~TidalTurbine()	152
4.12.3 Member Function Documentation	153
4.12.3.1 __handleKeyPressEvents()	153
4.12.3.2 __handleMouseButtonEvents()	153
4.12.3.3 __setUpTileImprovementSpriteAnimated()	154
4.12.3.4 draw()	154
4.12.3.5 processEvent()	155
4.12.3.6 processMessage()	155
4.13 TileImprovement Class Reference	155
4.13.1 Detailed Description	158
4.13.2 Constructor & Destructor Documentation	158
4.13.2.1 TileImprovement()	158
4.13.2.2 ~TileImprovement()	159
4.13.3 Member Function Documentation	159
4.13.3.1 __handleKeyPressEvents()	159
4.13.3.2 __handleMouseButtonEvents()	159
4.13.3.3 draw()	160
4.13.3.4 processEvent()	161
4.13.3.5 processMessage()	162

4.13.4 Member Data Documentation	162
4.13.4.1 assets_manager_ptr	162
4.13.4.2 credits	162
4.13.4.3 event_ptr	162
4.13.4.4 frame	163
4.13.4.5 game_phase	163
4.13.4.6 is_running	163
4.13.4.7 is_selected	163
4.13.4.8 just_built	163
4.13.4.9 message_hub_ptr	163
4.13.4.10 position_x	164
4.13.4.11 position_y	164
4.13.4.12 render_window_ptr	164
4.13.4.13 tile_improvement_sprite_animated	164
4.13.4.14 tile_improvement_sprite_static	164
4.13.4.15 tile_improvement_string	164
4.13.4.16 tile_improvement_type	165
4.14 WaveEnergyConverter Class Reference	165
4.14.1 Detailed Description	166
4.14.2 Constructor & Destructor Documentation	166
4.14.2.1 WaveEnergyConverter()	166
4.14.2.2 ~WaveEnergyConverter()	167
4.14.3 Member Function Documentation	167
4.14.3.1 __handleKeyPressEvents()	167
4.14.3.2 __handleMouseButtonEvents()	168
4.14.3.3 __setUpTileImprovementSpriteAnimated()	168
4.14.3.4 draw()	169
4.14.3.5 processEvent()	169
4.14.3.6 processMessage()	170
4.15 WindTurbine Class Reference	170
4.15.1 Detailed Description	171
4.15.2 Constructor & Destructor Documentation	172
4.15.2.1 WindTurbine()	172
4.15.2.2 ~WindTurbine()	172
4.15.3 Member Function Documentation	173
4.15.3.1 __handleKeyPressEvents()	173
4.15.3.2 __handleMouseButtonEvents()	173
4.15.3.3 __setUpTileImprovementSpriteAnimated()	174
4.15.3.4 draw()	174
4.15.3.5 processEvent()	175
4.15.3.6 processMessage()	175

5 File Documentation	177
5.1 header/ContextMenu.h File Reference	177
5.1.1 Detailed Description	178
5.1.2 Enumeration Type Documentation	178
5.1.2.1 ConsoleState	178
5.2 header/DieselGenerator.h File Reference	178
5.2.1 Detailed Description	179
5.3 header/EnergyStorageSystem.h File Reference	179
5.3.1 Detailed Description	180
5.4 header/ESC_core/AssetsManager.h File Reference	180
5.4.1 Detailed Description	181
5.5 header/ESC_core/constants.h File Reference	181
5.5.1 Detailed Description	183
5.5.2 Function Documentation	183
5.5.2.1 FOREST_GREEN()	184
5.5.2.2 LAKE_BLUE()	184
5.5.2.3 MENU_FRAME_GREY()	184
5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()	184
5.5.2.5 MONOCHROME_TEXT_AMBER()	184
5.5.2.6 MONOCHROME_TEXT_GREEN()	185
5.5.2.7 MONOCHROME_TEXT_RED()	185
5.5.2.8 MOUNTAINS_GREY()	185
5.5.2.9 OCEAN_BLUE()	185
5.5.2.10 PLAINS_YELLOW()	185
5.5.2.11 RESOURCE_CHIP_GREY()	186
5.5.2.12 VISUAL_SCREEN_FRAME_GREY()	186
5.5.3 Variable Documentation	186
5.5.3.1 BUILD_SETTLEMENT_COST	186
5.5.3.2 CLEAR_FOREST_COST	186
5.5.3.3 CLEAR_MOUNTAINS_COST	186
5.5.3.4 CLEAR_PLAINS_COST	187
5.5.3.5 CO2E_KG_PER_LITRE_DIESEL	187
5.5.3.6 DIESEL_GENERATOR_BUILD_COST	187
5.5.3.7 EMISSIONS_LIFETIME_LIMIT_TONNES	187
5.5.3.8 ENERGY_STORAGE_SYSTEM_BUILD_COST	187
5.5.3.9 FLOAT_TOLERANCE	187
5.5.3.10 FRAMES_PER_SECOND	188
5.5.3.11 GAME_CHANNEL	188
5.5.3.12 GAME_HEIGHT	188
5.5.3.13 GAME_STATE_CHANNEL	188
5.5.3.14 GAME_WIDTH	188
5.5.3.15 HEX_MAP_CHANNEL	188

5.5.3.16 NO_TILE_SELECTED_CHANNEL	189
5.5.3.17 RESOURCE_ASSESSMENT_COST	189
5.5.3.18 SECONDS_PER_FRAME	189
5.5.3.19 SECONDS_PER_MONTH	189
5.5.3.20 SECONDS_PER_YEAR	189
5.5.3.21 SOLAR_PV_BUILD_COST	189
5.5.3.22 SOLAR_PV_WATER_BUILD_MULTIPLIER	190
5.5.3.23 STARTING_CREDITS	190
5.5.3.24 STARTING_POPULATION	190
5.5.3.25 TIDAL_TURBINE_BUILD_COST	190
5.5.3.26 TILE_RESOURCE_CUMULATIVE_PROBABILITIES	190
5.5.3.27 TILE_SELECTED_CHANNEL	191
5.5.3.28 TILE_STATE_CHANNEL	191
5.5.3.29 TILE_TYPE_CUMULATIVE_PROBABILITIES	191
5.5.3.30 WAVE_ENERGY_CONVERTER_BUILD_COST	191
5.5.3.31 WIND_TURBINE_BUILD_COST	191
5.5.3.32 WIND_TURBINE_WATER_BUILD_MULTIPLIER	191
5.6 header/ESC_core/doxygen_cite.h File Reference	192
5.6.1 Detailed Description	192
5.7 header/ESC_core/includes.h File Reference	192
5.7.1 Detailed Description	193
5.8 header/ESC_core/MessageHub.h File Reference	193
5.8.1 Detailed Description	193
5.9 header/ESC_core/testing_utils.h File Reference	194
5.9.1 Detailed Description	195
5.9.2 Function Documentation	195
5.9.2.1 expectedErrorNotDetected()	195
5.9.2.2 printGold()	195
5.9.2.3 printGreen()	196
5.9.2.4 printRed()	196
5.9.2.5 testFloatEquals()	196
5.9.2.6 testGreaterThan()	197
5.9.2.7 testGreaterThanOrEqualTo()	197
5.9.2.8 testLessThan()	198
5.9.2.9 testLessThanOrEqualTo()	199
5.9.2.10 testTruth()	199
5.10 header/Game.h File Reference	200
5.10.1 Enumeration Type Documentation	201
5.10.1.1 GamePhase	201
5.11 header/HexMap.h File Reference	201
5.11.1 Detailed Description	202
5.12 header/HexTile.h File Reference	202

5.12.1 Detailed Description	203
5.12.2 Enumeration Type Documentation	204
5.12.2.1 TileResource	204
5.12.2.2 TileType	204
5.13 header/Settlement.h File Reference	205
5.13.1 Detailed Description	205
5.14 header/SolarPV.h File Reference	206
5.14.1 Detailed Description	206
5.15 header/TidalTurbine.h File Reference	207
5.15.1 Detailed Description	207
5.16 header/TileImprovement.h File Reference	208
5.16.1 Detailed Description	208
5.16.2 Enumeration Type Documentation	208
5.16.2.1 TileImprovementType	208
5.17 header/WaveEnergyConverter.h File Reference	209
5.17.1 Detailed Description	210
5.18 header/WindTurbine.h File Reference	210
5.18.1 Detailed Description	211
5.19 source/ContextMenu.cpp File Reference	211
5.19.1 Detailed Description	211
5.20 source/DieselGenerator.cpp File Reference	212
5.20.1 Detailed Description	212
5.21 source/EnergyStorageSystem.cpp File Reference	212
5.21.1 Detailed Description	212
5.22 source/ESC_core/AssetsManager.cpp File Reference	212
5.22.1 Detailed Description	213
5.23 source/ESC_core/MessageHub.cpp File Reference	213
5.23.1 Detailed Description	213
5.24 source/ESC_core/testing_utils.cpp File Reference	213
5.24.1 Detailed Description	214
5.24.2 Function Documentation	214
5.24.2.1 expectedErrorNotDetected()	214
5.24.2.2 printGold()	215
5.24.2.3 printGreen()	215
5.24.2.4 printRed()	215
5.24.2.5 testFloatEquals()	216
5.24.2.6 testGreaterThan()	216
5.24.2.7 testGreaterThanOrEqualTo()	217
5.24.2.8 testLessThan()	218
5.24.2.9 testLessThanOrEqualTo()	218
5.24.2.10 testTruth()	219
5.25 source/Game.cpp File Reference	219

5.25.1 Detailed Description	220
5.26 source/HexMap.cpp File Reference	220
5.26.1 Detailed Description	220
5.27 source/HexTile.cpp File Reference	220
5.27.1 Detailed Description	221
5.28 source/main.cpp File Reference	221
5.28.1 Detailed Description	221
5.28.2 Function Documentation	221
5.28.2.1 constructRenderWindow()	221
5.28.2.2 loadAssets()	222
5.28.2.3 main()	223
5.29 source/Settlement.cpp File Reference	224
5.29.1 Detailed Description	224
5.30 source/SolarPV.cpp File Reference	224
5.30.1 Detailed Description	225
5.31 source/TidalTurbine.cpp File Reference	225
5.31.1 Detailed Description	225
5.32 source/TileImprovement.cpp File Reference	225
5.32.1 Detailed Description	225
5.33 source/WaveEnergyConverter.cpp File Reference	226
5.33.1 Detailed Description	226
5.34 source/WindTurbine.cpp File Reference	226
5.34.1 Detailed Description	226
Bibliography	227
Index	229

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AssetsManager	7
ContextMenu	19
Game	49
HexMap	64
HexTile	86
Message	130
MessageHub	131
TileImprovement	155
DieselGenerator	37
EnergyStorageSystem	44
Settlement	138
SolarPV	145
TidalTurbine	150
WaveEnergyConverter	165
WindTurbine	170

Chapter 2

Class Index

2.1 Class List

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

AssetsManager	A class which manages visual and sound assets	7
ContextMenu	A class which defines a context menu for the game	19
DieselGenerator	A settlement class (child class of TileImprovement)	37
EnergyStorageSystem	A settlement class (child class of TileImprovement)	44
Game	A class which acts as the central class for the game, by containing all other classes and implementing the game loop	49
HexMap	A class which defines a hex map of hex tiles	64
HexTile	A class which defines a hex tile of the hex map	86
Message	A structure which defines a standard message format	130
MessageHub	A class which acts as a central hub for inter-object message traffic	131
Settlement	A settlement class (child class of TileImprovement)	138
SolarPV	A settlement class (child class of TileImprovement)	145
TidalTurbine	A settlement class (child class of TileImprovement)	150
TileImprovement	A base class for the tile improvement hierarchy	155
WaveEnergyConverter	A settlement class (child class of TileImprovement)	165
WindTurbine	A settlement class (child class of TileImprovement)	170

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

header/ ContextMenu.h	
Header file for the ContextMenu class	177
header/ DieselGenerator.h	
Header file for the DieselGenerator class	178
header/ EnergyStorageSystem.h	
Header file for the EnergyStorageSystem class	179
header/ Game.h	200
header/ HexMap.h	
Header file for the HexMap class	201
header/ HexTile.h	
Header file for the Game class	202
header/ Settlement.h	
Header file for the Settlement class	205
header/ SolarPV.h	
Header file for the SolarPV class	206
header/ TidalTurbine.h	
Header file for the TidalTurbine class	207
header/ TileImprovement.h	
Header file for the TileImprovement class	208
header/ WaveEnergyConverter.h	
Header file for the WaveEnergyConverter class	209
header/ WindTurbine.h	
Header file for the WindTurbine class	210
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	180
header/ESC_core/ constants.h	
Header file for various constants	181
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	192
header/ESC_core/ includes.h	
Header file for various includes	192
header/ESC_core/ MessageHub.h	
Header file for the MessageHub class	193
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	194

source/ ContextMenu.cpp	Implementation file for the ContextMenu class	211
source/ DieselGenerator.cpp	Implementation file for the DieselGenerator class	212
source/ EnergyStorageSystem.cpp	Implementation file for the EnergyStorageSystem class	212
source/ Game.cpp	Implementation file for the Game class	219
source/ HexMap.cpp	Implementation file for the HexMap class	220
source/ HexTile.cpp	Implementation file for the HexTile class	220
source/ main.cpp	Implementation file for main() for Road To Zero	221
source/ Settlement.cpp	Implementation file for the Settlement class	224
source/ SolarPV.cpp	Implementation file for the SolarPV class	224
source/ TidalTurbine.cpp	Implementation file for the TidalTurbine class	225
source/ TileImprovement.cpp	Implementation file for the TileImprovement class	225
source/ WaveEnergyConverter.cpp	Implementation file for the WaveEnergyConverter class	226
source/ WindTurbine.cpp	Implementation file for the WindTurbine class	226
source/ESC_core/ AssetsManager.cpp	Implementation file for the AssetsManager class	212
source/ESC_core/ MessageHub.cpp	Implementation file for the MessageHub class	213
source/ESC_core/ testing_utils.cpp	Implementation file for various testing utilities	213

Chapter 4

Class Documentation

4.1 AssetsManager Class Reference

A class which manages visual and sound assets.

```
#include <AssetsManager.h>
```

Public Member Functions

- [AssetsManager](#) (void)
Constructor for the [AssetsManager](#) class.
- void [loadFont](#) (std::string, std::string)
Method to load a font and insert it into the font map.
- void [loadTexture](#) (std::string, std::string)
Method to load a texture and insert it into the texture map.
- void [loadSound](#) (std::string, std::string)
Method to load a sound and insert it into the sound map. Automatically creates a corresponding sf::SoundBuffer.
- void [loadTrack](#) (std::string, std::string)
Method to load a track (sf::Music) and insert it into the track map.
- sf::Font * [getFont](#) (std::string)
Method to get font associated with given font key.
- sf::Texture * [getTexture](#) (std::string)
Method to get texture associated with given texture key.
- sf::SoundBuffer * [getSoundBuffer](#) (std::string)
Method to get soundbuffer associated with given sound key.
- sf::Sound * [getSound](#) (std::string)
Method to get sound associated with given sound key.
- void [playTrack](#) (void)
Method to play the current track.
- void [pauseTrack](#) (void)
Method to pause the current track.
- void [stopTrack](#) (void)
Method to stop the current track.
- void [nextTrack](#) (void)
Method to advance to the next track. Wraps around if the end of the track map is reached.

- void [previousTrack](#) (void)
Method to return to the previous track. Wraps around if the beginning of the track map is reached.
- std::string [getCurrentTrackKey](#) (void)
Method to get track key for current track.
- sf::SoundSource::Status [getTrackStatus](#) (void)
Method to get the status of the current track.
- void [clear](#) (void)
Method to clear all loaded assets.
- [~AssetsManager](#) (void)
Destructor for the [AssetsManager](#) class.

Public Attributes

- std::map< std::string, sf::Font * > [font_map](#)
A map of pointers to loaded fonts.
- std::map< std::string, sf::Texture * > [texture_map](#)
A map of pointers to loaded textures.
- std::map< std::string, sf::SoundBuffer * > [soundbuffer_map](#)
A map of pointers to sound buffers.
- std::map< std::string, sf::Sound * > [sound_map](#)
A map of pointers to loaded sounds.
- std::map< std::string, sf::Music * >::iterator [current_track](#)
A map iterator which corresponds to the current track (i.e., the track currently being played).
- std::map< std::string, sf::Music * > [track_map](#)
A map of pointers to opened tracks (i.e. sf::Music).

Private Member Functions

- void [__loadSoundBuffer](#) (std::string, std::string)
Helper method to load a soundbuffer and insert it into the soundbuffer map. Should only be called by [loadSound\(\)](#), to create an sf::SoundBuffer corresponding to the loaded sf::Sound.

4.1.1 Detailed Description

A class which manages visual and sound assets.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AssetsManager()

```
AssetsManager::AssetsManager (
    void )
```

Constructor for the [AssetsManager](#) class.

```
110 {
111     //...
112
113     std::cout << "AssetsManager constructed at " << this << std::endl;
114
115     return;
116 } /* AssetsManager() */
```


4.1.2.2 ~AssetsManager()

```
AssetsManager::~AssetsManager (
    void )
```

Destructor for the [AssetsManager](#) class.

```
739 {
740     this->clear();
741
742     std::cout << "AssetsManager at " << this << " destroyed" << std::endl;
743
744     return;
745 } /* ~AssetsManager() */
```

4.1.3 Member Function Documentation

4.1.3.1 __loadSoundBuffer()

```
void AssetsManager::__loadSoundBuffer (
    std::string path_2_sound,
    std::string sound_key ) [private]
```

Helper method to load a soundbuffer and insert it into the soundbuffer map. Should only be called by [loadSound\(\)](#), to create an `sf::SoundBuffer` corresponding to the loaded `sf::Sound`.

Parameters

<i>path_2_sound</i>	A path (either relative or absolute) to the sound file.
<i>sound_key</i>	A key associated with the sound (for indexing into the soundbuffer map).

```
47 {
48     // 1. check key, throw error if already in use
49     if (this->soundbuffer_map.count(sound_key) > 0) {
50         std::string error_str = "ERROR AssetsManager::__loadSoundBuffer() sound key ";
51         error_str += sound_key;
52         error_str += " is already in use";
53
54         this->clear();
55
56         #ifdef _WIN32
57             std::cout << error_str << std::endl;
58         #endif /* _WIN32 */
59
60         throw std::runtime_error(error_str);
61     }
62
63
64     // 2. load from file, throw error on fail
65     sf::SoundBuffer* soundbuffer_ptr = new sf::SoundBuffer();
66
67     if (not soundbuffer_ptr->loadFromFile(path_2_sound)) {
68         std::string error_str = "ERROR AssetsManager::__loadSoundBuffer() could not load ";
69         error_str += "soundbuffer at ";
70         error_str += path_2_sound;
71
72         this->clear();
73
74         #ifdef _WIN32
75             std::cout << error_str << std::endl;
76         #endif /* _WIN32 */
77
78         throw std::runtime_error(error_str);
79     }
80
81 }
```

```

82 // 3. insert into soundbuffer map
83 this->soundbuffer_map.insert(
84     std::pair<std::string, sf::SoundBuffer*>(sound_key, soundbuffer_ptr)
85 );
86
87 std::cout << "SoundBuffer " << sound_key << " inserted into soundbuffer map" <<
88     std::endl;
89
90 return;
91 } /* __loadSoundBuffer() */

```

4.1.3.2 clear()

```

void AssetsManager::clear (
    void )

```

Method to clear all loaded assets.

```

646 {
647     // 1. clear fonts
648     std::map<std::string, sf::Font*>::iterator font_iter;
649     for (
650         font_iter = this->font_map.begin();
651         font_iter != this->font_map.end();
652         font_iter++
653     ) {
654         delete font_iter->second;
655
656         std::cout << "Font " << font_iter->first << " deleted from font map" <<
657             std::endl;
658     }
659     this->font_map.clear();
660
661     // 2. clear textures
662     std::map<std::string, sf::Texture*>::iterator texture_iter;
663     for (
664         texture_iter = this->texture_map.begin();
665         texture_iter != this->texture_map.end();
666         texture_iter++
667     ) {
668         delete texture_iter->second;
669
670         std::cout << "Texture " << texture_iter->first << " deleted from texture map" <<
671             std::endl;
672     }
673     this->texture_map.clear();
674
675     // 3. clear sound buffers
676     std::map<std::string, sf::SoundBuffer*>::iterator soundbuffer_iter;
677     for (
678         soundbuffer_iter = this->soundbuffer_map.begin();
679         soundbuffer_iter != this->soundbuffer_map.end();
680         soundbuffer_iter++
681     ) {
682         delete soundbuffer_iter->second;
683
684         std::cout << "SoundBuffer " << soundbuffer_iter->first <<
685             " deleted from soundbuffer map" << std::endl;
686     }
687     this->soundbuffer_map.clear();
688
689     // 4. clear sounds
690     std::map<std::string, sf::Sound*>::iterator sound_iter;
691     for (
692         sound_iter = this->sound_map.begin();
693         sound_iter != this->sound_map.end();
694         sound_iter++
695     ) {
696         sound_iter->second->stop();
697         delete sound_iter->second;
698
699         std::cout << "Sound " << sound_iter->first << " deleted from sound map" <<
700             std::endl;
701     }
702     this->sound_map.clear();
703
704 }

```

```

707
708 // 5. clear tracks
709 std::map<std::string, sf::Music*>::iterator track_iter;
710 for (
711     track_iter = this->track_map.begin();
712     track_iter != this->track_map.end();
713     track_iter++
714 ) {
715     track_iter->second->stop();
716     delete track_iter->second;
717
718     std::cout << "Track " << track_iter->first << " deleted from track map" <<
719         std::endl;
720 }
721 this->track_map.clear();
722
723 return;
724 } /* clear() */

```

4.1.3.3 getCurrentTrackKey()

```

std::string AssetsManager::getCurrentTrackKey (
    void )

```

Method to get track key for current track.

Returns

The track key for the current track.

```

610 {
611     return this->current_track->first;
612 } /* getCurrentTrackKey() */

```

4.1.3.4 getFont()

```

sf::Font * AssetsManager::getFont (
    std::string font_key )

```

Method to get font associated with given font key.

Parameters

<i>font_key</i>	A key associated with the font (for indexing into the font map).
-----------------	--

Returns

A pointer to the corresponding font.

```

351 {
352     // 1. check key, throw error if not found
353     if (this->font_map.count(font_key) <= 0) {
354         std::string error_str = "ERROR AssetsManager::getFont() font key ";
355         error_str += font_key;
356         error_str += " is not contained in font map";
357
358         this->clear();
359
360         #ifdef _WIN32

```

```

361         std::cout << error_str << std::endl;
362     #endif /* _WIN32 */
363
364     throw std::runtime_error(error_str);
365 }
366
367 return this->font_map[font_key];
368 } /* getFont() */

```

4.1.3.5 getSound()

```

sf::Sound * AssetsManager::getSound (
    std::string sound_key )

```

Method to get sound associated with given sound key.

Parameters

<i>sound_key</i>	A key associated with the sound (for indexing into the sound map).
------------------	--

Returns

A pointer to the corresponding sound.

```

461 {
462     // 1. check key, throw error if not found
463     if (this->sound_map.count(sound_key) <= 0) {
464         std::string error_str = "ERROR AssetsManager::getSound() sound key ";
465         error_str += sound_key;
466         error_str += " is not contained in sound map";
467
468         this->clear();
469
470         #ifdef _WIN32
471             std::cout << error_str << std::endl;
472         #endif /* _WIN32 */
473
474         throw std::runtime_error(error_str);
475     }
476
477     return this->sound_map[sound_key];
478 } /* getSound() */

```

4.1.3.6 getSoundBuffer()

```

sf::SoundBuffer * AssetsManager::getSoundBuffer (
    std::string sound_key )

```

Method to get soundbuffer associated with given sound key.

Parameters

<i>sound_key</i>	A key associated with the soundbuffer (for indexing into the soundbuffer map).
------------------	--

Returns

A pointer to the corresponding soundbuffer.

```

425 {
426     // 1. check key, throw error if not found
427     if (this->soundbuffer_map.count(sound_key) <= 0) {
428         std::string error_str = "ERROR AssetsManager::getSoundBuffer() sound key ";
429         error_str += sound_key;
430         error_str += " is not contained in soundbuffer map";
431
432         this->clear();
433
434         #ifdef _WIN32
435             std::cout << error_str << std::endl;
436         #endif /* _WIN32 */
437
438         throw std::runtime_error(error_str);
439     }
440
441     return this->soundbuffer_map[sound_key];
442 } /* getSoundBuffer() */

```

4.1.3.7 getTexture()

```

sf::Texture * AssetsManager::getTexture (
    std::string texture_key )

```

Method to get texture associated with given texture key.

Parameters

<i>texture_key</i>	A key associated with the texture (for indexing into the texture map).
--------------------	--

Returns

A pointer to the corresponding texture.

```

388 {
389     // 1. check key, throw error if not found
390     if (this->texture_map.count(texture_key) <= 0) {
391         std::string error_str = "ERROR AssetsManager::getTexture() texture key ";
392         error_str += texture_key;
393         error_str += " is not contained in texture map";
394
395         this->clear();
396
397         #ifdef _WIN32
398             std::cout << error_str << std::endl;
399         #endif /* _WIN32 */
400
401         throw std::runtime_error(error_str);
402     }
403
404     return this->texture_map[texture_key];
405 } /* getTexture() */

```

4.1.3.8 getTrackStatus()

```

sf::SoundSource::Status AssetsManager::getTrackStatus (
    void )

```

Method to get the status of the current track.

Returns

The status of the current track.

```

629 {
630     return this->current_track->second->getStatus();
631 } /* getTrackStatus */

```

4.1.3.9 loadFont()

```

void AssetsManager::loadFont (
    std::string path_2_font,
    std::string font_key )

```

Method to load a font and insert it into the font map.

Parameters

<i>path_2_font</i>	A path (either relative or absolute) to the font file.
<i>font_key</i>	A key associated with the font (for indexing into the font map).

```

135 {
136     // 1. check key, throw error if already in use
137     if (this->font_map.count(font_key) > 0) {
138         std::string error_str = "ERROR AssetsManager::loadFont() font key ";
139         error_str += font_key;
140         error_str += " is already in use";
141
142         this->clear();
143
144         #ifdef _WIN32
145             std::cout << error_str << std::endl;
146         #endif /* _WIN32 */
147
148         throw std::runtime_error(error_str);
149     }
150
151     // 2. load from file, throw error on fail
152     sf::Font* font_ptr = new sf::Font();
153
154     if (not font_ptr->loadFromFile(path_2_font)) {
155         std::string error_str = "ERROR AssetsManager::loadFont() could not load ";
156         error_str += "font at ";
157         error_str += path_2_font;
158
159         this->clear();
160
161         #ifdef _WIN32
162             std::cout << error_str << std::endl;
163         #endif /* _WIN32 */
164
165         throw std::runtime_error(error_str);
166     }
167
168     // 3. insert into font map
169     this->font_map.insert(std::pair<std::string, sf::Font*>(font_key, font_ptr));
170
171     std::cout << "Font " << font_key << " inserted into font map" << std::endl;
172
173     return;
174 } /* loadFont() */

```

4.1.3.10 loadSound()

```

void AssetsManager::loadSound (

```

```
std::string path_2_sound,
std::string sound_key )
```

Method to load a sound and insert it into the sound map. Automatically creates a corresponding sf::SoundBuffer.

Parameters

<i>path_2_sound</i>	A path (either relative or absolute) to the sound file.
<i>sound_key</i>	A key associated with the sound (for indexing into the sound map).

```
259 {
260     // 1. create an associated sf::SoundBuffer
261     this->__loadSoundBuffer(path_2_sound, sound_key);
262
263     // 2. associate sf::Sound with sf::SoundBuffer
264     sf::Sound* sound_ptr = new sf::Sound();
265     sound_ptr->setBuffer(*(this->soundbuffer_map[sound_key]));
266
267     // 3. insert into sound map
268     this->sound_map.insert(std::pair<std::string, sf::Sound*>(sound_key, sound_ptr));
269
270     std::cout << "Sound " << sound_key << " inserted into sound map" << std::endl;
271
272     return;
273 } /* loadSound() */
```

4.1.3.11 loadTexture()

```
void AssetsManager::loadTexture (
    std::string path_2_texture,
    std::string texture_key )
```

Method to load a texture and insert it into the texture map.

Parameters

<i>path_2_texture</i>	A path (either relative or absolute) to the texture file.
<i>texture_key</i>	A key associated with the texture (for indexing into the texture map).

```
196 {
197     // 1. check key, throw error if already in use
198     if (this->texture_map.count(texture_key) > 0) {
199         std::string error_str = "ERROR AssetsManager::loadTexture() texture key ";
200         error_str += texture_key;
201         error_str += " is already in use";
202
203         this->clear();
204
205         #ifdef _WIN32
206             std::cout << error_str << std::endl;
207         #endif /* _WIN32 */
208
209         throw std::runtime_error(error_str);
210     }
211
212     // 2. load from file, throw error on fail
213     sf::Texture* texture_ptr = new sf::Texture();
214
215     if (not texture_ptr->loadFromFile(path_2_texture)) {
216         std::string error_str = "ERROR AssetsManager::loadTexture() could not load ";
217         error_str += "texture at ";
218         error_str += path_2_texture;
219
220         this->clear();
221
222         #ifdef _WIN32
223             std::cout << error_str << std::endl;
224         #endif
```

```

225         #endif /* _WIN32 */
226
227         throw std::runtime_error(error_str);
228     }
229
230
231     // 3. insert into texture map
232     this->texture_map.insert(
233         std::pair<std::string, sf::Texture*>(texture_key, texture_ptr)
234     );
235
236     std::cout << "Texture " << texture_key << " inserted into texture map" << std::endl;
237
238     return;
239 } /* loadTexture() */

```

4.1.3.12 loadTrack()

```

void AssetsManager::loadTrack (
    std::string path_2_track,
    std::string track_key )

```

Method to load a track (sf::Music) and insert it into the track map.

Parameters

<i>path_2_track</i>	A path (either relative or absolute) to the track file.
<i>track_key</i>	A key associated with the track (for indexing into the track map).

```

292 {
293     // 1. check key, throw error if already in use
294     if (this->track_map.count(track_key) > 0) {
295         std::string error_str = "ERROR AssetsManager::loadTrack() track key ";
296         error_str += track_key;
297         error_str += " is already in use";
298
299         this->clear();
300
301         #ifdef _WIN32
302             std::cout << error_str << std::endl;
303         #endif /* _WIN32 */
304
305         throw std::runtime_error(error_str);
306     }
307
308     // 2. open from file, throw error on fail
309     sf::Music* track_ptr = new sf::Music();
310
311     if (not track_ptr->openFromFile(path_2_track)) {
312         std::string error_str = "ERROR AssetsManager::loadTrack() could not open ";
313         error_str += "track at ";
314         error_str += path_2_track;
315
316         this->clear();
317
318         #ifdef _WIN32
319             std::cout << error_str << std::endl;
320         #endif /* _WIN32 */
321
322         throw std::runtime_error(error_str);
323     }
324
325     // 3. insert into track map
326     this->track_map.insert(std::pair<std::string, sf::Music*>(track_key, track_ptr));
327     this->current_track = this->track_map.begin();
328
329     std::cout << "Track " << track_key << " inserted into track map" << std::endl;
330
331     return;
332 } /* loadTrack() */

```


4.1.3.13 nextTrack()

```
void AssetsManager::nextTrack (
    void )
```

Method to advance to the next track. Wraps around if the end of the track map is reached.

```
551 {
552     // 1. stop current track
553     this->stopTrack();
554
555     // 2. increment current track
556     this->current_track++;
557
558     // 3. handle wrap around
559     if (this->current_track == this->track_map.end()) {
560         this->current_track = this->track_map.begin();
561     }
562
563     return;
564 } /* nextTrack() */
```

4.1.3.14 pauseTrack()

```
void AssetsManager::pauseTrack (
    void )
```

Method to pause the current track.

```
512 {
513     this->current_track->second->pause();
514
515     return;
516 } /* pauseTrack() */
```

4.1.3.15 playTrack()

```
void AssetsManager::playTrack (
    void )
```

Method to play the current track.

```
493 {
494     this->current_track->second->play();
495
496     return;
497 } /* playTrack() */
```

4.1.3.16 previousTrack()

```
void AssetsManager::previousTrack (
    void )
```

Method to return to the previous track. Wraps around if the beginning of the track map is reached.

```
580 {
581     // 1. stop current track
582     this->stopTrack();
583
584     // 2. handle wrap around
585     if (this->current_track == this->track_map.begin()) {
586         this->current_track = this->track_map.end();
587     }
588
589     // 3. decrement current track
590     this->current_track--;
591
592     return;
593 } /* previousTrack() */
```

4.1.3.17 stopTrack()

```
void AssetsManager::stopTrack (
    void )
```

Method to stop the current track.

```
531 {
532     this->current_track->second->stop();
533
534     return;
535 } /* stopTrack() */
```

4.1.4 Member Data Documentation

4.1.4.1 current_track

```
std::map<std::string, sf::Music*>::iterator AssetsManager::current_track
```

A map iterator which corresponds to the current track (i.e., the track currently being played).

4.1.4.2 font_map

```
std::map<std::string, sf::Font*> AssetsManager::font_map
```

A map of pointers to loaded fonts.

4.1.4.3 sound_map

```
std::map<std::string, sf::Sound*> AssetsManager::sound_map
```

A map of pointers to loaded sounds.

4.1.4.4 soundbuffer_map

```
std::map<std::string, sf::SoundBuffer*> AssetsManager::soundbuffer_map
```

A map of pointers to sound buffers.

4.1.4.5 texture_map

```
std::map<std::string, sf::Texture*> AssetsManager::texture_map
```

A map of pointers to loaded textures.

4.1.4.6 track_map

```
std::map<std::string, sf::Music*> AssetsManager::track_map
```

A map of pointers to opened tracks (i.e. sf::Music).

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

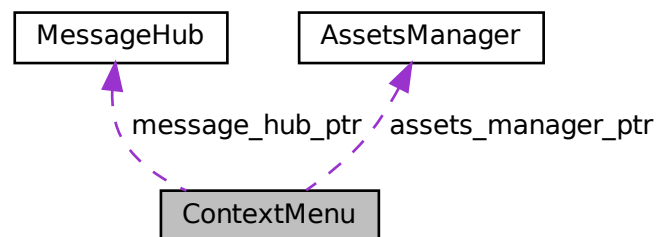
- header/ESC_core/[AssetsManager.h](#)
- source/ESC_core/[AssetsManager.cpp](#)

4.2 ContextMenu Class Reference

A class which defines a context menu for the game.

```
#include <ContextMenu.h>
```

Collaboration diagram for ContextMenu:



Public Member Functions

- [ContextMenu](#) (sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [ContextMenu](#) class.
- void [processEvent](#) (void)
Method to processEvent [ContextMenu](#). To be called once per event.
- void [processMessage](#) (void)
Method to processMessage [ContextMenu](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- [~ContextMenu](#) (void)
Destructor for the [ContextMenu](#) class.

Public Attributes

- [ConsoleState console_state](#)
The current state of the console screen.
- bool [console_string_changed](#)
Boolean which indicates if console string just changed.
- bool [game_menu_up](#)
Indicates whether or not the game menu is up.
- size_t [console_substring_idx](#)
The current final index of the console string draw.
- int [frame](#)
The current frame of this object.
- double [position_x](#)
The position of the object.
- double [position_y](#)
The position of the object.
- std::string [console_string](#)
The string to be printed to the console screen.
- sf::RectangleShape [menu_frame](#)
The frame of the context menu.
- sf::RectangleShape [visual_screen](#)
The context menu screen for visuals.
- sf::ConvexShape [visual_screen_frame_top](#)
The top framing of the visual screen.
- sf::ConvexShape [visual_screen_frame_left](#)
The left framing of the visual screen.
- sf::ConvexShape [visual_screen_frame_bottom](#)
The bottom framing of the visual screen.
- sf::ConvexShape [visual_screen_frame_right](#)
The right framing of the visual screen.
- sf::RectangleShape [console_screen](#)
The context menu console screen (for animated text output).
- sf::ConvexShape [console_screen_frame_top](#)
The top framing of the console screen.
- sf::ConvexShape [console_screen_frame_left](#)
The left framing of the console screen.
- sf::ConvexShape [console_screen_frame_bottom](#)
The bottom framing of the console screen.
- sf::ConvexShape [console_screen_frame_right](#)
The right framing of the console screen.

Private Member Functions

- void [__setUpMenuFrame](#) (void)
Helper method to set up context menu frame (drawable).
- void [__setUpVisualScreen](#) (void)
Helper method to set up context menu visual screen (drawable).
- void [__setUpVisualScreenFrame](#) (void)
Helper method to set up framing for context menu visual screen (drawable).
- void [__drawVisualScreenFrame](#) (void)

- Helper method to draw visual screen frame.*
- void [__setUpConsoleScreen](#) (void)
- Helper method to set up context menu console screen (drawable).*
- void [__setUpConsoleScreenFrame](#) (void)
- Helper method to set up framing for context menu console screen (drawable).*
- void [__drawConsoleScreenFrame](#) (void)
- Helper method to draw console screen frame.*
- void [__setConsoleState](#) (ConsoleState)
- Helper method to set state of console screen and update string if necessary.*
- void [__setConsoleString](#) (void)
- Helper method to set console string depending on console state.*
- void [__drawConsoleText](#) (void)
- Helper method to draw animated text to context menu console screen.*
- void [__handleKeyPressEvents](#) (void)
- Helper method to handle key press events.*
- void [__handleMouseButtonEvents](#) (void)
- Helper method to handle mouse button events.*
- void [__sendQuitGameMessage](#) (void)
- Helper method to format and send a quit game message.*
- void [__sendRestartGameMessage](#) (void)
- Helper method to format and send a restart game message.*

Private Attributes

- sf::Event * [event_ptr](#)
- A pointer to the event class.*
- sf::RenderWindow * [render_window_ptr](#)
- A pointer to the render window.*
- [AssetsManager](#) * [assets_manager_ptr](#)
- A pointer to the assets manager.*
- [MessageHub](#) * [message_hub_ptr](#)
- A pointer to the message hub.*

4.2.1 Detailed Description

A class which defines a context menu for the game.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ContextMenu()

```
ContextMenu::ContextMenu (
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [ContextMenu](#) class.

Parameters

<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```

815 {
816     // 1. set attributes
817
818     // 1.1. private
819     this->event_ptr = event_ptr;
820     this->render_window_ptr = render_window_ptr;
821
822     this->assets_manager_ptr = assets_manager_ptr;
823     this->message_hub_ptr = message_hub_ptr;
824
825     // 1.2. public
826     this->console_state = ConsoleState :: NONE_STATE;
827     this->__setConsoleState(ConsoleState :: READY);
828
829     this->console_string_changed = true;
830     this->game_menu_up = false;
831
832     this->frame = 0;
833
834     this->position_x = GAME_WIDTH;
835     this->position_y = 0;
836
837     // 2. set up and position drawable attributes
838     this->__setUpMenuFrame();
839     this->__setUpVisualScreen();
840     this->__setUpVisualScreenFrame();
841     this->__setUpConsoleScreen();
842     this->__setUpConsoleScreenFrame();
843
844     std::cout << "ContextMenu constructed at " << this << std::endl;
845
846     return;
847 } /* ContextMenu() */

```

4.2.2.2 ~ContextMenu()

```

ContextMenu::~~ContextMenu (
    void )

```

Destructor for the [ContextMenu](#) class.

```

997 {
998     std::cout << "ContextMenu at " << this << " destroyed" << std::endl;
999
1000     return;
1001 } /* ~ContextMenu() */

```

4.2.3 Member Function Documentation

4.2.3.1 __drawConsoleScreenFrame()

```

void ContextMenu::__drawConsoleScreenFrame (
    void ) [private]

```

Helper method to draw console screen frame.

```

433 {
434     this->render_window_ptr->draw(this->console_screen_frame_top);
435     this->render_window_ptr->draw(this->console_screen_frame_left);
436     this->render_window_ptr->draw(this->console_screen_frame_bottom);
437     this->render_window_ptr->draw(this->console_screen_frame_right);
438
439     return;
440 } /* __drawContextScreenFrame() */

```

4.2.3.2 __drawConsoleText()

```

void ContextMenu::__drawConsoleText (
    void ) [private]

```

Helper method to draw animated text to context menu console screen.

```

556 {
557     // 1. set up console text (drawable)
558     sf::Text console_text;
559
560     if (this->console_string_changed) {
561         this->assets_manager_ptr->getSound("console string print")->play();
562
563         console_text.setString(this->console_string.substr(0, this->console_substring_idx));
564
565         this->console_substring_idx++;
566
567         while (
568             (this->console_string.substr(0, this->console_substring_idx).back() == ' ') or
569             (this->console_string.substr(0, this->console_substring_idx).back() == '\n')
570         ) {
571             this->console_substring_idx++;
572
573             if (this->console_substring_idx >= this->console_string.size()) {
574                 break;
575             }
576         }
577
578         if (this->console_substring_idx >= this->console_string.size()) {
579             this->console_string_changed = false;
580         }
581     }
582
583     else {
584         console_text.setString(this->console_string);
585     }
586
587     console_text.setFont(*(this->assets_manager_ptr->getFont("Glass_TTY_VT220")));
588     console_text.setCharacterSize(16);
589     console_text.setFillColor(MONOCROME_TEXT_GREEN);
590
591     console_text.setPosition(
592         this->position_x - 50 - 300 + 16,
593         this->position_y + GAME_HEIGHT - 50 - 340 + 16
594     );
595
596
597     // 2. draw console text
598     this->render_window_ptr->draw(console_text);
599
600
601     // 3. assemble and draw blinking console cursor
602     if ((this->frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
603         sf::RectangleShape console_cursor(sf::Vector2f(10, 16));
604
605         console_cursor.setFillColor(MONOCROME_TEXT_GREEN);
606
607         console_cursor.setPosition(
608             console_text.getPosition().x,
609             console_text.getPosition().y + console_text.getLocalBounds().height + 10
610         );
611
612         this->render_window_ptr->draw(console_cursor);
613     }
614
615     // 4. updating frame count if console is in menu state
616     if (this->console_state == ConsoleState::MENU) {
617         std::string frame_count_string = "FRAME: ";
618         frame_count_string += std::to_string(this->frame);

```

```

619
620     sf::Text frame_count_text (
621         frame_count_string,
622         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
623         16
624     );
625
626     frame_count_text.setFillColor(MONOCHROME_TEXT_GREEN);
627
628     frame_count_text.setPosition(
629         console_text.getPosition().x,
630         console_text.getPosition().y + console_text.getLocalBounds().height - 10
631     );
632
633     this->render_window_ptr->draw(frame_count_text);
634 }
635
636 return;
637 } /* __drawConsoleText() */

```

4.2.3.3 __drawVisualScreenFrame()

```

void ContextMenu::__drawVisualScreenFrame (
    void ) [private]

```

Helper method to draw visual screen frame.

```

208 {
209     this->render_window_ptr->draw(this->visual_screen_frame_top);
210     this->render_window_ptr->draw(this->visual_screen_frame_left);
211     this->render_window_ptr->draw(this->visual_screen_frame_bottom);
212     this->render_window_ptr->draw(this->visual_screen_frame_right);
213
214     return;
215 } /* __drawVisualScreenFrame() */

```

4.2.3.4 __handleKeyPressEvents()

```

void ContextMenu::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

652 {
653     switch (this->event_ptr->key.code) {
654         case (sf::Keyboard::Escape): {
655             if (this->console_state == ConsoleState :: MENU) {
656                 this->__setConsoleState(ConsoleState :: READY);
657             }
658
659             else {
660                 this->__setConsoleState(ConsoleState :: MENU);
661             }
662
663             break;
664         }
665
666         case (sf::Keyboard::Q): {
667             if (this->console_state == ConsoleState :: MENU) {
668                 this->__sendQuitGameMessage();
669             }
670         }
671
672         case (sf::Keyboard::R): {
673             if (this->console_state == ConsoleState :: MENU) {
674                 this->__sendRestartGameMessage();
675             }
676         }
677     }
678 }
679

```



```

680
681         default: {
682             // do nothing!
683
684             break;
685         }
686     }
687
688     return;
689 } /* __handleKeyPressEvents() */

```

4.2.3.5 __handleMouseButtonEvents()

```

void ContextMenu::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

704 {
705     switch (this->event_ptr->mouseButton.button) {
706         case (sf::Mouse::Left): {
707             //...
708
709             break;
710         }
711
712         case (sf::Mouse::Right): {
713             //...
714
715             break;
716         }
717
718         default: {
719             // do nothing!
720
721             break;
722         }
723     }
724 }
725
726
727     return;
728 } /* __handleMouseButtonEvents() */

```

4.2.3.6 __sendQuitGameMessage()

```

void ContextMenu::__sendQuitGameMessage (
    void ) [private]

```

Helper method to format and send a quit game message.

```

743 {
744     Message quit_game_message;
745
746     quit_game_message.channel = GAME_CHANNEL;
747     quit_game_message.subject = "quit game";
748
749     this->message_hub_ptr->sendMessage(quit_game_message);
750
751     std::cout << "Quit game message sent by " << this << std::endl;
752     return;
753 } /* __sendQuitGameMessage() */

```

4.2.3.7 __sendRestartGameMessage()

```
void ContextMenu::__sendRestartGameMessage (
    void ) [private]
```

Helper method to format and send a restart game message.

```
768 {
769     Message restart_game_message;
770
771     restart_game_message.channel = GAME_CHANNEL;
772     restart_game_message.subject = "restart game";
773
774     this->message_hub_ptr->sendMessage(restart_game_message);
775
776     std::cout << "Restart game message sent by " << this << std::endl;
777     return;
778 } /* __sendRestartGameMessage() */
```

4.2.3.8 __setConsoleState()

```
void ContextMenu::__setConsoleState (
    ConsoleState console_state ) [private]
```

Helper method to set state of console screen and update string if necessary.

Parameters

<i>console_state</i>	The state (ConsoleState) to set the console to.
----------------------	---

```
457 {
458     // 1. if no change, do nothing
459     if (this->console_state == console_state) {
460         return;
461     }
462
463     // 2. update console state, set console string accordingly
464     this->console_state = console_state;
465     this->__setConsoleString();
466
467     return;
468 } /* __setConsoleState() */
```

4.2.3.9 __setConsoleString()

```
void ContextMenu::__setConsoleString (
    void ) [private]
```

Helper method to set console string depending on console state.

```
483 {
484     this->console_string_changed = true;
485     this->console_substring_idx = 0;
486
487     this->console_string.clear();
488
489     switch (this->console_state) {
490     case (ConsoleState :: MENU): {
491         // 32 char x 17 line console "-----\n";
492         this->console_string = "          **** MENU ****          \n";
493         this->console_string += "          \n";
494         this->console_string += "[R]:  RESTART          \n";
495         this->console_string += "          \n";
496         this->console_string += "[TAB]: TOGGLE RESOURCE OVERLAY \n";
497     }
```

```

497         this->console_string += "[T]:  TUTORIAL          \n";
498         this->console_string += "                  \n";
499         this->console_string += "                  \n";
500         this->console_string += "                  \n";
501         this->console_string += "                  \n";
502         this->console_string += "                  \n";
503         this->console_string += "                  \n";
504         this->console_string += "                  \n";
505         this->console_string += "[Q]:    QUIT          \n";
506         this->console_string += "[ESC]:  CLOSE MENU    \n";
507         this->console_string += "                  \n";
508
509         break;
510     }
511
512
513     case (ConsoleState :: TILE): {
514         // take console string from tile state message
515
516         break;
517     }
518
519
520     default: {
521         //          32 char x 17 line console "-----\n";
522         this->console_string = "    **** RTZ 64 CONTEXT V12 **** \n";
523         this->console_string += "                  \n";
524         this->console_string += "64K RAM SYSTEM  38911 BYTES FREE\n";
525         this->console_string += "                  \n";
526         this->console_string += "[TAB]:  TOGGLE RESOURCE OVERLAY \n";
527         this->console_string += "                  \n";
528         this->console_string += "[ESC]:           MENU          \n";
529         this->console_string += "[LEFT CLICK]:  TILE INFO/OPTIONS\n";
530         this->console_string += "[RIGHT CLICK]: CLEAR SELECTION  \n";
531         this->console_string += "                  \n";
532         this->console_string += "[ENTER]:  END TURN            \n";
533         this->console_string += "                  \n";
534         this->console_string += "READY.                        ";
535
536         break;
537     }
538 }
539
540 return;
541 } /* __setConsoleString() */

```

4.2.3.10 __setUpConsoleScreen()

```

void ContextMenu::__setUpConsoleScreen (
    void ) [private]

```

Helper method to set up context menu console screen (drawable).

```

230 {
231     this->console_screen.setSize(sf::Vector2f(300, 340));
232     this->console_screen.setOrigin(300, 340);
233     this->console_screen.setPosition(
234         this->position_x - 50,
235         this->position_y + GAME_HEIGHT - 50
236     );
237     this->console_screen.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
238
239     return;
240 } /* __setUpConsoleScreen() */

```

4.2.3.11 __setUpConsoleScreenFrame()

```

void ContextMenu::__setUpConsoleScreenFrame (
    void ) [private]

```

Helper method to set up framing for context menu console screen (drawable).

```

255 {
256     int n_points = 4;
257
258     // 1. top framing
259     this->console_screen_frame_top.setPointCount(n_points);
260
261     this->console_screen_frame_top.setPoint(
262         0,
263         sf::Vector2f(
264             this->position_x - 50,
265             this->position_y + GAME_HEIGHT - 50 - 340
266         )
267     );
268     this->console_screen_frame_top.setPoint(
269         1,
270         sf::Vector2f(
271             this->position_x - 50 + 16,
272             this->position_y + GAME_HEIGHT - 50 - 340 - 16
273         )
274     );
275     this->console_screen_frame_top.setPoint(
276         2,
277         sf::Vector2f(
278             this->position_x - 350 - 16,
279             this->position_y + GAME_HEIGHT - 50 - 340 - 16
280         )
281     );
282     this->console_screen_frame_top.setPoint(
283         3,
284         sf::Vector2f(
285             this->position_x - 350,
286             this->position_y + GAME_HEIGHT - 50 - 340
287         )
288     );
289
290     this->console_screen_frame_top.setFillColors(VISUAL_SCREEN_FRAME_GREY);
291
292     this->console_screen_frame_top.setOutlineThickness(2);
293     this->console_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
294
295     this->console_screen_frame_top.move(0, -2);
296
297
298     // 2. left framing
299     this->console_screen_frame_left.setPointCount(n_points);
300
301     this->console_screen_frame_left.setPoint(
302         0,
303         sf::Vector2f(
304             this->position_x - 350,
305             this->position_y + GAME_HEIGHT - 50 - 340
306         )
307     );
308     this->console_screen_frame_left.setPoint(
309         1,
310         sf::Vector2f(
311             this->position_x - 350 - 16,
312             this->position_y + GAME_HEIGHT - 50 - 340 - 16
313         )
314     );
315     this->console_screen_frame_left.setPoint(
316         2,
317         sf::Vector2f(
318             this->position_x - 350 - 16,
319             this->position_y + GAME_HEIGHT - 50 + 16
320         )
321     );
322     this->console_screen_frame_left.setPoint(
323         3,
324         sf::Vector2f(
325             this->position_x - 350,
326             this->position_y + GAME_HEIGHT - 50
327         )
328     );
329
330     this->console_screen_frame_left.setFillColors(VISUAL_SCREEN_FRAME_GREY);
331
332     this->console_screen_frame_left.setOutlineThickness(2);
333     this->console_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
334
335     this->console_screen_frame_left.move(-2, 0);
336
337
338     // 3. bottom framing
339     this->console_screen_frame_bottom.setPointCount(n_points);
340

```

```

341     this->console_screen_frame_bottom.setPoint(
342         0,
343         sf::Vector2f(
344             this->position_x - 350,
345             this->position_y + GAME_HEIGHT - 50
346         )
347     );
348     this->console_screen_frame_bottom.setPoint(
349         1,
350         sf::Vector2f(
351             this->position_x - 350 - 16,
352             this->position_y + GAME_HEIGHT - 50 + 16
353         )
354     );
355     this->console_screen_frame_bottom.setPoint(
356         2,
357         sf::Vector2f(
358             this->position_x - 50 + 16,
359             this->position_y + GAME_HEIGHT - 50 + 16
360         )
361     );
362     this->console_screen_frame_bottom.setPoint(
363         3,
364         sf::Vector2f(
365             this->position_x - 50,
366             this->position_y + GAME_HEIGHT - 50
367         )
368     );
369     this->console_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
370
371     this->console_screen_frame_bottom.setOutlineThickness(2);
372     this->console_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
373
374     this->console_screen_frame_bottom.move(0, 2);
375
376
377     // 4. right framing
378     this->console_screen_frame_right.setPointCount(n_points);
379
380     this->console_screen_frame_right.setPoint(
381         0,
382         sf::Vector2f(
383             this->position_x - 50,
384             this->position_y + GAME_HEIGHT - 50
385         )
386     );
387
388     this->console_screen_frame_right.setPoint(
389         1,
390         sf::Vector2f(
391             this->position_x - 50 + 16,
392             this->position_y + GAME_HEIGHT - 50 + 16
393         )
394     );
395     this->console_screen_frame_right.setPoint(
396         2,
397         sf::Vector2f(
398             this->position_x - 50 + 16,
399             this->position_y + GAME_HEIGHT - 50 - 340 - 16
400         )
401     );
402     this->console_screen_frame_right.setPoint(
403         3,
404         sf::Vector2f(
405             this->position_x - 50,
406             this->position_y + GAME_HEIGHT - 50 - 340
407         )
408     );
409     this->console_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
410
411     this->console_screen_frame_right.setOutlineThickness(2);
412     this->console_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
413
414     this->console_screen_frame_right.move(2, 0);
415
416     return;
417 } /* __setUpConsoleScreenFrame() */
418 }

```

4.2.3.12 __setUpMenuFrame()

```
void ContextMenu::__setUpMenuFrame (
```

```
void ) [private]
```

Helper method to set up context menu frame (drawable).

```
34 {
35     this->menu_frame.setSize(sf::Vector2f(400, GAME_HEIGHT));
36     this->menu_frame.setOrigin(400, 0);
37     this->menu_frame.setPosition(this->position_x, this->position_y);
38     this->menu_frame.setFillColor(MENU_FRAME_GREY);
39
40     return;
41 } /* __setUpMenuFrame() */
```

4.2.3.13 __setUpVisualScreen()

```
void ContextMenu::__setUpVisualScreen (
    void ) [private]
```

Helper method to set up context menu visual screen (drawable).

```
56 {
57     this->visual_screen.setSize(sf::Vector2f(300, 300));
58     this->visual_screen.setOrigin(300, 0);
59     this->visual_screen.setPosition(this->position_x - 50, this->position_y + 50);
60     this->visual_screen.setFillColor(MONochrome_SCREEN_BACKGROUND);
61
62     return;
63 } /* __setUpVisualScreen() */
```

4.2.3.14 __setUpVisualScreenFrame()

```
void ContextMenu::__setUpVisualScreenFrame (
    void ) [private]
```

Helper method to set up framing for context menu visual screen (drawable).

```
78 {
79     int n_points = 4;
80
81     // 1. top framing
82     this->visual_screen_frame_top.setPointCount(n_points);
83
84     this->visual_screen_frame_top.setPoint(
85         0,
86         sf::Vector2f(this->position_x - 50, this->position_y + 50)
87     );
88     this->visual_screen_frame_top.setPoint(
89         1,
90         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
91     );
92     this->visual_screen_frame_top.setPoint(
93         2,
94         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
95     );
96     this->visual_screen_frame_top.setPoint(
97         3,
98         sf::Vector2f(this->position_x - 350, this->position_y + 50)
99     );
100
101     this->visual_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
102
103     this->visual_screen_frame_top.setOutlineThickness(2);
104     this->visual_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
105
106     this->visual_screen_frame_top.move(0, -2);
107
108
109     // 2. left framing
110     this->visual_screen_frame_left.setPointCount(n_points);
111
112     this->visual_screen_frame_left.setPoint(
```

```

113         0,
114         sf::Vector2f(this->position_x - 350, this->position_y + 50)
115     );
116     this->visual_screen_frame_left.setPoint(
117         1,
118         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
119     );
120     this->visual_screen_frame_left.setPoint(
121         2,
122         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
123     );
124     this->visual_screen_frame_left.setPoint(
125         3,
126         sf::Vector2f(this->position_x - 350, this->position_y + 350)
127     );
128
129     this->visual_screen_frame_left.setFill-color(VISUAL_SCREEN_FRAME_GREY);
130
131     this->visual_screen_frame_left.setOutlineThickness(2);
132     this->visual_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
133
134     this->visual_screen_frame_left.move(-2, 0);
135
136
137     // 3. bottom framing
138     this->visual_screen_frame_bottom.setPointCount(n_points);
139
140     this->visual_screen_frame_bottom.setPoint(
141         0,
142         sf::Vector2f(this->position_x - 350, this->position_y + 350)
143     );
144     this->visual_screen_frame_bottom.setPoint(
145         1,
146         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
147     );
148     this->visual_screen_frame_bottom.setPoint(
149         2,
150         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
151     );
152     this->visual_screen_frame_bottom.setPoint(
153         3,
154         sf::Vector2f(this->position_x - 50, this->position_y + 350)
155     );
156
157     this->visual_screen_frame_bottom.setFill-color(VISUAL_SCREEN_FRAME_GREY);
158
159     this->visual_screen_frame_bottom.setOutlineThickness(2);
160     this->visual_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
161
162     this->visual_screen_frame_bottom.move(0, 2);
163
164
165     // 4. right framing
166     this->visual_screen_frame_right.setPointCount(n_points);
167
168     this->visual_screen_frame_right.setPoint(
169         0,
170         sf::Vector2f(this->position_x - 50, this->position_y + 350)
171     );
172     this->visual_screen_frame_right.setPoint(
173         1,
174         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
175     );
176     this->visual_screen_frame_right.setPoint(
177         2,
178         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
179     );
180     this->visual_screen_frame_right.setPoint(
181         3,
182         sf::Vector2f(this->position_x - 50, this->position_y + 50)
183     );
184
185     this->visual_screen_frame_right.setFill-color(VISUAL_SCREEN_FRAME_GREY);
186
187     this->visual_screen_frame_right.setOutlineThickness(2);
188     this->visual_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
189
190     this->visual_screen_frame_right.move(2, 0);
191
192     return;
193 } /* __setUpVisualScreenFrame() */

```

4.2.3.15 draw()

```
void ContextMenu::draw (
    void )
```

Method to draw the hex tile to the render window. To be called once per frame.

```
967 {
968     // 1. menu frame
969     this->render_window_ptr->draw(this->menu_frame);
970
971     // 2. visual screen
972     this->render_window_ptr->draw(this->visual_screen);
973     this->__drawVisualScreenFrame();
974
975     // 3. console screen
976     this->render_window_ptr->draw(this->console_screen);
977     this->__drawConsoleScreenFrame();
978     this->__drawConsoleText();
979
980     this->frame++;
981     return;
982 } /* draw() */
```

4.2.3.16 processEvent()

```
void ContextMenu::processEvent (
    void )
```

Method to processEvent [ContextMenu](#). To be called once per event.

```
862 {
863     if (this->event_ptr->type == sf::Event::KeyPressed) {
864         this->__handleKeyPressEvents();
865     }
866
867     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
868         this->__handleMouseButtonEvents();
869     }
870
871     return;
872 } /* processEvent() */
```

4.2.3.17 processMessage()

```
void ContextMenu::processMessage (
    void )
```

Method to processMessage [ContextMenu](#). To be called once per message.

```
887 {
888     switch (this->console_state) {
889         case (ConsoleState :: TILE): {
890             // process no tile selected
891             if (not this->message_hub_ptr->isEmpty(NO_TILE_SELECTED_CHANNEL)) {
892                 Message no_tile_selected_message = this->message_hub_ptr->receiveMessage(
893                     NO_TILE_SELECTED_CHANNEL
894                 );
895
896                 if (no_tile_selected_message.subject == "no tile selected") {
897                     this->__setConsoleState(ConsoleState :: READY);
898
899                     std::cout << "No tile selected message received by " << this <<
900                         std::endl;
901                     this->message_hub_ptr->popMessage(NO_TILE_SELECTED_CHANNEL);
902                 }
903             }
904
905             // process tile state
```



```

906         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
907             Message tile_state_message = this->message_hub_ptr->receiveMessage(
908                 TILE_STATE_CHANNEL
909             );
910
911             if (tile_state_message.subject == "tile state") {
912                 this->console_string = tile_state_message.string_payload["console string"];
913
914                 this->console_string_changed = true;
915                 this->console_substring_idx = 0;
916
917                 std::cout << "Tile state message received by " << this << std::endl;
918                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
919             }
920         }
921
922         // process tile selected (subsequent left clicks causing program to hang)
923         if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
924             this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
925         }
926
927         break;
928     }
929
930     default: {
931         // process tile selected
932         if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
933             Message tile_selected_message = this->message_hub_ptr->receiveMessage(
934                 TILE_SELECTED_CHANNEL
935             );
936
937             if (tile_selected_message.subject == "tile selected") {
938                 this->__setConsoleState(ConsoleState :: TILE);
939
940                 std::cout << "Tile selected message received by " << this <<
941                     std::endl;
942                 this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
943             }
944         }
945
946         break;
947     }
948 }
949
950 return;
951 } /* processMessage() */

```

4.2.4 Member Data Documentation

4.2.4.1 assets_manager_ptr

`AssetsManager*` ContextMenu::assets_manager_ptr [private]

A pointer to the assets manager.

4.2.4.2 console_screen

`sf::RectangleShape` ContextMenu::console_screen

The context menu console screen (for animated text output).

4.2.4.3 console_screen_frame_bottom

```
sf::ConvexShape ContextMenu::console_screen_frame_bottom
```

The bottom framing of the console screen.

4.2.4.4 console_screen_frame_left

```
sf::ConvexShape ContextMenu::console_screen_frame_left
```

The left framing of the console screen.

4.2.4.5 console_screen_frame_right

```
sf::ConvexShape ContextMenu::console_screen_frame_right
```

The right framing of the console screen.

4.2.4.6 console_screen_frame_top

```
sf::ConvexShape ContextMenu::console_screen_frame_top
```

The top framing of the console screen.

4.2.4.7 console_state

```
ConsoleState ContextMenu::console_state
```

The current state of the console screen.

4.2.4.8 console_string

```
std::string ContextMenu::console_string
```

The string to be printed to the console screen.

4.2.4.9 console_string_changed

```
bool ContextMenu::console_string_changed
```

Boolean which indicates if console string just changed.

4.2.4.10 console_substring_idx

```
size_t ContextMenu::console_substring_idx
```

The current final index of the console string draw.

4.2.4.11 event_ptr

```
sf::Event* ContextMenu::event_ptr [private]
```

A pointer to the event class.

4.2.4.12 frame

```
int ContextMenu::frame
```

The current frame of this object.

4.2.4.13 game_menu_up

```
bool ContextMenu::game_menu_up
```

Indicates whether or not the game menu is up.

4.2.4.14 menu_frame

```
sf::RectangleShape ContextMenu::menu_frame
```

The frame of the context menu.

4.2.4.15 message_hub_ptr

```
MessageHub* ContextMenu::message_hub_ptr [private]
```

A pointer to the message hub.

4.2.4.16 position_x

```
double ContextMenu::position_x
```

The position of the object.

4.2.4.17 position_y

```
double ContextMenu::position_y
```

The position of the object.

4.2.4.18 render_window_ptr

```
sf::RenderWindow* ContextMenu::render_window_ptr [private]
```

A pointer to the render window.

4.2.4.19 visual_screen

```
sf::RectangleShape ContextMenu::visual_screen
```

The context menu screen for visuals.

4.2.4.20 visual_screen_frame_bottom

```
sf::ConvexShape ContextMenu::visual_screen_frame_bottom
```

The bottom framing of the visual screen.

4.2.4.21 visual_screen_frame_left

```
sf::ConvexShape ContextMenu::visual_screen_frame_left
```

The left framing of the visual screen.

4.2.4.22 visual_screen_frame_right

```
sf::ConvexShape ContextMenu::visual_screen_frame_right
```

The right framing of the visual screen.

4.2.4.23 visual_screen_frame_top

```
sf::ConvexShape ContextMenu::visual_screen_frame_top
```

The top framing of the visual screen.

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

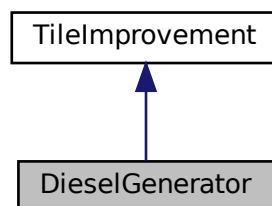
- header/[ContextMenu.h](#)
- source/[ContextMenu.cpp](#)

4.3 DieselGenerator Class Reference

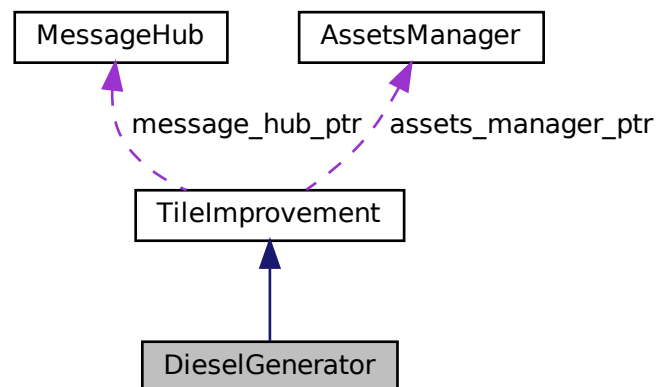
A settlement class (child class of [TileImprovement](#)).

```
#include <DieselGenerator.h>
```

Inheritance diagram for DieselGenerator:



Collaboration diagram for DieselGenerator:



Public Member Functions

- [DieselGenerator](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [DieselGenerator](#) class.
- void [processEvent](#) (void)
Method to process [DieselGenerator](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [DieselGenerator](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~DieselGenerator](#) (void)
Destructor for the [DieselGenerator](#) class.

Public Attributes

- bool [skip_smoke_processing](#)
A boolean which indicates whether or not to skip smoke processing.
- double [smoke_da](#)
The per frame delta in smoke particle alpha value.
- double [smoke_dx](#)
The per frame delta in smoke particle x position.
- double [smoke_dy](#)
The per frame delta in smoke particle y position.
- double [smoke_prob](#)
The probability of spawning a new smoke prob in any given frame.
- std::list< sf::Sprite > [smoke_sprite_list](#)
A list of smoke sprite (for chimney animation).

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.3.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DieselGenerator()

```
DieselGenerator::DieselGenerator (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [DieselGenerator](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
178 :
179 TileImprovement (
180     position_x,
181     position_y,
182     event_ptr,
183     render_window_ptr,
184     assets_manager_ptr,
185     message_hub_ptr
186 )
```

```

187 {
188     // 1. set attributes
189
190     // 1.1. private
191     //...
192
193     // 1.2. public
194     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
195
196     this->is_running = false;
197     this->skip_smoke_processing = true;
198
199     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
200     this->smoke_dx = 5 * SECONDS_PER_FRAME;
201     this->smoke_dy = -10 * SECONDS_PER_FRAME;
202     this->smoke_prob = 8 * SECONDS_PER_FRAME;
203
204     this->smoke_sprite_list = {};
205
206     this->tile_improvement_string = "DIESEL GEN";
207
208     this->__setUpTileImprovementSpriteAnimated();
209
210     std::cout << "DieselGenerator constructed at " << this << std::endl;
211
212     return;
213 } /* DieselGenerator() */

```

4.3.2.2 ~DieselGenerator()

```

DieselGenerator::~~DieselGenerator (
    void ) [virtual]

```

Destructor for the [DieselGenerator](#) class.

```

322 {
323     std::cout << "DieselGenerator at " << this << " destroyed" << std::endl;
324
325     return;
326 } /* ~DieselGenerator() */

```

4.3.3 Member Function Documentation

4.3.3.1 __handleKeyPressEvents()

```

void DieselGenerator::__handleKeyPressEvents (
    void ) [private], [virtual]

```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```

80 {
81     switch (this->event_ptr->key.code) {
82         //...
83
84         default: {
85             // do nothing!
86
87             break;
88         }
89     }
90
91     return;
92 } /* __handleKeyPressEvents() */

```


4.3.3.2 __handleMouseButtonEvents()

```
void DieselGenerator::__handleMouseButtonEvents (
    void ) [private], [virtual]
```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
108 {
109     switch (this->event_ptr->mouseButton.button) {
110         case (sf::Mouse::Left): {
111             //...
112             break;
113         }
114
115
116         case (sf::Mouse::Right): {
117             //...
118             break;
119         }
120
121         default: {
122             // do nothing!
123             break;
124         }
125     }
126     return;
127 } /* __handleMouseButtonEvents() */
```

4.3.3.3 __setUpTileImprovementSpriteAnimated()

```
void DieselGenerator::__setUpTileImprovementSpriteAnimated (
    void ) [private]
```

Helper method to set up tile improvement sprite (static).

```
34 {
35     sf::Sprite diesel_generator_sheet (
36         *(this->assets_manager_ptr->getTexture("diesel generator"))
37     );
38
39     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
40
41     for (int i = 0; i < n_elements; i++) {
42         this->tile_improvement_sprite_animated.push_back(
43             sf::Sprite(
44                 *(this->assets_manager_ptr->getTexture("diesel generator")),
45                 sf::IntRect(0, i * 64, 64, 64)
46             )
47         );
48
49         this->tile_improvement_sprite_animated.back().setOrigin(
50             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
51             this->tile_improvement_sprite_animated.back().getLocalBounds().height
52         );
53
54         this->tile_improvement_sprite_animated.back().setPosition(
55             this->position_x,
56             this->position_y - 32
57         );
58
59         this->tile_improvement_sprite_animated.back().setColor(
60             sf::Color(255, 255, 255, 0)
61         );
62     }
63     return;
64 } /* __setUpTileImprovementSpriteAnimated() */
```

4.3.3.4 draw()

```
void DieselGenerator::draw (
    void ) [virtual]
```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```
273 {
274     // 1. if just built, call base method and return
275     if (this->just_built) {
276         TileImprovement :: draw();
277     }
278     return;
279 }
280
281
282 // 1. draw first element of animated sprite
283 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
284
285
286 // 2. draw second element of animated sprite
287 if (this->is_running) {
288     //...
289 }
290
291 else {
292     //...
293 }
294
295 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
296
297
298 // 3. draw smoke effects
299 if (this->is_running) {
300     //...
301 }
302
303 //...
304
305 this->frame++;
306 return;
307 } /* draw() */
```

4.3.3.5 processEvent()

```
void DieselGenerator::processEvent (
    void ) [virtual]
```

Method to process [DieselGenerator](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```
228 {
229     if (this->event_ptr->type == sf::Event::KeyPressed) {
230         this->__handleKeyPressEvents();
231     }
232
233     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
234         this->__handleMouseButtonEvents();
235     }
236
237     return;
238 } /* processEvent() */
```

4.3.3.6 processMessage()

```
void DieselGenerator::processMessage (
    void ) [virtual]
```

Method to process [DieselGenerator](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
253 {
254     // ...
255
256     return;
257 } /* processMessage() */
```

4.3.4 Member Data Documentation

4.3.4.1 skip_smoke_processing

```
bool DieselGenerator::skip_smoke_processing
```

A boolean which indicates whether or not to skip smoke processing.

4.3.4.2 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.3 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.4 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.5 smoke_prob

```
double DieselGenerator::smoke_prob
```

The probability of spawning a new smoke prob in any given frame.

4.3.4.6 smoke_sprite_list

```
std::list<sf::Sprite> DieselGenerator::smoke_sprite_list
```

A list of smoke sprite (for chimney animation).

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

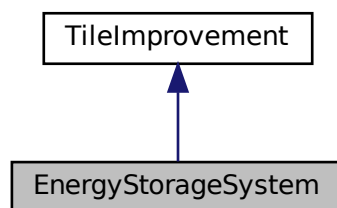
- header/[DieselGenerator.h](#)
- source/[DieselGenerator.cpp](#)

4.4 EnergyStorageSystem Class Reference

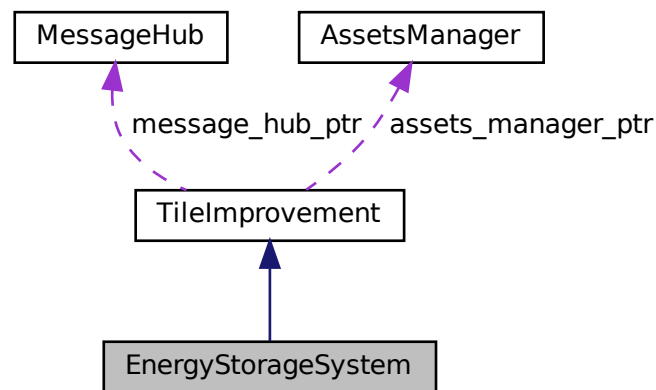
A settlement class (child class of [TileImprovement](#)).

```
#include <EnergyStorageSystem.h>
```

Inheritance diagram for EnergyStorageSystem:



Collaboration diagram for EnergyStorageSystem:



Public Member Functions

- [EnergyStorageSystem](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [EnergyStorageSystem](#) class.
- void [processEvent](#) (void)
Method to process [EnergyStorageSystem](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [EnergyStorageSystem](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~EnergyStorageSystem](#) (void)
Destructor for the [EnergyStorageSystem](#) class.

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.4.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 EnergyStorageSystem()

```
EnergyStorageSystem::EnergyStorageSystem (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [EnergyStorageSystem](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
167 :
168 TileImprovement (
169     position_x,
170     position_y,
171     event_ptr,
172     render_window_ptr,
173     assets_manager_ptr,
174     message_hub_ptr
175 )
176 {
177     // 1. set attributes
178
179     // 1.1. private
180     //...
181
182     // 1.2. public
183     this->tile_improvement_type = TileImprovementType :: ENERGY_STORAGE_SYSTEM;
184
185     this->is_running = false;
186
187     this->tile_improvement_string = "ENERGY STORAGE";
188
189     this->__setUpTileImprovementSpriteStatic();
190
191     std::cout << "EnergyStorageSystem constructed at " << this << std::endl;
192
193     return;
194 } /* EnergyStorageSystem() */
```

4.4.2.2 ~EnergyStorageSystem()

```
EnergyStorageSystem::~EnergyStorageSystem (
    void ) [virtual]
```

Destructor for the [EnergyStorageSystem](#) class.

```
283 {
284     std::cout << "EnergyStorageSystem at " << this << " destroyed" << std::endl;
285
286     return;
287 } /* ~EnergyStorageSystem() */
```

4.4.3 Member Function Documentation

4.4.3.1 __handleKeyPressEvents()

```
void EnergyStorageSystem::__handleKeyPressEvents (
    void ) [private], [virtual]
```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```
69 {
70     switch (this->event_ptr->key.code) {
71         //...
72
73         default: {
74             // do nothing!
75
76             break;
77         }
78     }
79 }
80
81 return;
82 } /* __handleKeyPressEvents() */
```

4.4.3.2 __handleMouseButtonEvents()

```
void EnergyStorageSystem::__handleMouseButtonEvents (
    void ) [private], [virtual]
```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
97 {
98     switch (this->event_ptr->mouseButton.button) {
99         case (sf::Mouse::Left): {
100             //...
101
102             break;
103         }
104
105         case (sf::Mouse::Right): {
106             //...
107
108             break;
109         }
110
111         default: {
112             // do nothing!
113
114             break;
115         }
116     }
117 }
118
119 return;
120 } /* __handleMouseButtonEvents() */
```

4.4.3.3 __setUpTileImprovementSpriteStatic()

```
void EnergyStorageSystem::__setUpTileImprovementSpriteStatic (
    void ) [private]
```

Helper method to set up tile improvement sprite (static).

```
34 {
35     this->tile_improvement_sprite_static.setTexture(
36         *(this->assets_manager_ptr->getTexture("energy storage system"))
37     );
38
39     this->tile_improvement_sprite_static.setOrigin(
40         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
41         this->tile_improvement_sprite_static.getLocalBounds().height
42     );
43
44     this->tile_improvement_sprite_static.setPosition(
45         this->position_x,
46         this->position_y - 32
47     );
48
49     this->tile_improvement_sprite_static.setColor(
50         sf::Color(255, 255, 255, 0)
51     );
52
53     return;
54 } /* __setUpTileImprovementSpriteStatic() */
```

4.4.3.4 draw()

```
void EnergyStorageSystem::draw (
    void ) [virtual]
```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```
254 {
255     // 1. if just built, call base method and return
256     if (this->just_built) {
257         TileImprovement::draw();
258
259         return;
260     }
261
262     // 1. draw static sprite
263     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
264
265     this->frame++;
266     return;
267 } /* draw() */
```

4.4.3.5 processEvent()

```
void EnergyStorageSystem::processEvent (
    void ) [virtual]
```

Method to process [EnergyStorageSystem](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```
209 {
210     if (this->event_ptr->type == sf::Event::KeyPressed) {
211         this->__handleKeyPressEvents();
212     }
213
214     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
215         this->__handleMouseButtonEvents();
216     }
217
218     return;
219 } /* processEvent() */
```


4.4.3.6 processMessage()

```
void EnergyStorageSystem::processMessage (
    void ) [virtual]
```

Method to process [EnergyStorageSystem](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
234 {
235     //...
236
237     return;
238 } /* processMessage() */
```

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

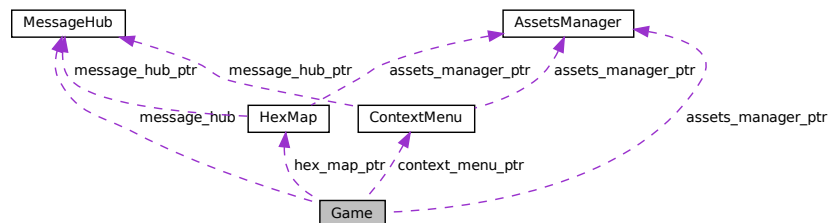
- header/[EnergyStorageSystem.h](#)
- source/[EnergyStorageSystem.cpp](#)

4.5 Game Class Reference

A class which acts as the central class for the game, by containing all other classes and implementing the game loop.

```
#include <Game.h>
```

Collaboration diagram for Game:



Public Member Functions

- [Game](#) (sf::RenderWindow *, [AssetsManager](#) *)
Constructor for the [Game](#) class.
- bool [run](#) (void)
Method to run game (defines game loop).
- [~Game](#) (void)
Destructor for the [Game](#) class.

Public Attributes

- [GamePhase](#) `game_phase`
The current phase of the game.
- `bool` [quit_game](#)
Boolean indicating whether to quit (true) or create a new [Game](#) instance (false).
- `bool` [game_loop_broken](#)
Boolean indicating whether or not the game loop is broken.
- `bool` [show_frame_clock_overlay](#)
Boolean indicating whether or not to show frame and clock overlay.
- `unsigned long long int` [frame](#)
The current frame of the game.
- `double` [time_since_start_s](#)
The time elapsed [s] since the start of the game.
- `int` [year](#)
Current game year.
- `int` [month](#)
Current game month.
- `int` [population](#)
Current population.
- `int` [credits](#)
Current balance of credits.
- `int` [demand_MWh](#)
Current energy demand [MWh].
- `int` [cumulative_emissions_tonnes](#)
Cumulative emissions [tonnes] (1 tonne = 1000 kg).
- `int` [turn](#) = 0
The current game turn.
- `sf::Clock` [clock](#)
The game clock.
- `sf::Event` [event](#)
The game events class.
- [MessageHub](#) [message_hub](#)
The message hub (for inter-object message traffic).
- [HexMap](#) * [hex_map_ptr](#)
Pointer to the hex map (defines game world).
- [ContextMenu](#) * [context_menu_ptr](#)
Pointer to the context menu.

Private Member Functions

- `void` [__toggleFrameClockOverlay](#) (void)
Helper method to toggle frame clock overlay.
- `void` [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- `void` [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- `void` [__processEvent](#) (void)
Helper method to process [Game](#). To be called once per event.
- `void` [__processMessage](#) (void)

- Helper method to process [Game](#). To be called once per message.*
- void [__sendGameStateMessage](#) (void)
Helper method to format and send a game state message.
- void [__insufficientCreditsAlarm](#) (void)
Helper method to sound and display and insufficient credits alarm.
- void [__drawFrameClockOverlay](#) (void)
Helper method to draw frame clock overlay.
- void [__drawHUD](#) (void)
Helper method to heads-up display (HUD).
- void [__draw](#) (void)
Helper method to draw game to the render window. To be called once per frame.

Private Attributes

- sf::RenderWindow * [render_window_ptr](#)
A pointer to the render window.
- [AssetsManager](#) * [assets_manager_ptr](#)
A pointer to the assets manager.

4.5.1 Detailed Description

A class which acts as the central class for the game, by containing all other classes and implementing the game loop.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Game()

```
Game::Game (
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr )
```

Constructor for the [Game](#) class.

```
662 {
663     // 1. set attributes
664
665     // 1.1. private
666     this->render_window_ptr = render_window_ptr;
667
668     this->assets_manager_ptr = assets_manager_ptr;
669
670     // 1.2. public
671     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
672
673     this->quit_game = false;
674     this->game_loop_broken = false;
675     this->show_frame_clock_overlay = false;
676
677     this->frame = 0;
678     this->time_since_start_s = 0;
679
680     double seconds_since_epoch = time(NULL);
681     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
682
683     this->year = 1970 + (int)years_since_epoch;
```

```

684     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
685
686     this->population = 0;
687     this->credits = STARTING_CREDITS;
688     this->demand_MWh = 0;
689     this->cumulative_emissions_tonnes = 0;
690
691     this->hex_map_ptr = new HexMap(
692         6,
693         &(this->event),
694         this->render_window_ptr,
695         this->assets_manager_ptr,
696         &(this->message_hub)
697     );
698
699     this->context_menu_ptr = new ContextMenu(
700         &(this->event),
701         this->render_window_ptr,
702         this->assets_manager_ptr,
703         &(this->message_hub)
704     );
705
706     // 2. add message channel(s)
707     this->message_hub.addChannel(GAME_CHANNEL);
708     this->message_hub.addChannel(GAME_STATE_CHANNEL);
709
710     std::cout << "Game constructed at " << this << std::endl;
711
712     return;
713 } /* Game() */

```

4.5.2.2 ~Game()

```

Game::~Game (
    void )

```

Destructor for the [Game](#) class.

```

790 {
791     // 1. clean up attributes
792     delete this->hex_map_ptr;
793     delete this->context_menu_ptr;
794
795     std::cout << "Game at " << this << " destroyed" << std::endl;
796
797     return;
798 } /* ~Game() */

```

4.5.3 Member Function Documentation

4.5.3.1 __draw()

```

void Game::__draw (
    void ) [private]

```

Helper method to draw game to the render window. To be called once per frame.

```

629 {
630     this->__drawHUD();
631
632     if (this->show_frame_clock_overlay) {
633         this->__drawFrameClockOverlay();
634     }
635
636     return;
637 } /* draw() */

```

4.5.3.2 __drawFrameClockOverlay()

```
void Game::__drawFrameClockOverlay (
    void ) [private]
```

Helper method to draw frame clock overlay.

```
455 {
456     std::string frame_clock_string = "FRAME: ";
457     frame_clock_string += std::to_string(this->frame);
458     frame_clock_string += "\nTIME SINCE START [s]: ";
459     frame_clock_string += std::to_string(this->time_since_start_s);
460
461     sf::Text frame_clock_text(
462         frame_clock_string,
463         *(this->assets_manager_ptr->getFont("DroidSansMono")),
464         16
465     );
466
467     sf::RectangleShape frame_clock_backing(
468         sf::Vector2f(
469             1.02 * frame_clock_text.getLocalBounds().width,
470             1.20 * frame_clock_text.getLocalBounds().height
471         )
472     );
473     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
474
475     this->render_window_ptr->draw(frame_clock_backing);
476     this->render_window_ptr->draw(frame_clock_text);
477
478     return;
479 } /* __drawFrameClockOverlay() */
```

4.5.3.3 __drawHUD()

```
void Game::__drawHUD (
    void ) [private]
```

Helper method to heads-up display (HUD).

```
494 {
495     // 1. first line (top)
496     std::string HUD_string = "YEAR: ";
497     HUD_string += std::to_string(this->year);
498
499     HUD_string += "    MONTH: ";
500     HUD_string += std::to_string(this->month);
501
502     HUD_string += "    POPULATION: ";
503     HUD_string += std::to_string(this->population);
504
505     HUD_string += "    CREDITS: ";
506     HUD_string += std::to_string(this->credits);
507     HUD_string += " K";
508
509     HUD_string += "    CURRENT DEMAND: ";
510     HUD_string += std::to_string(this->demand_MWh);
511     HUD_string += " MWh";
512
513     sf::Text HUD_text(
514         HUD_string,
515         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
516         16
517     );
518
519     HUD_text.setPosition(
520         (800 - HUD_text.getLocalBounds().width) / 2,
521         8
522     );
523
524     HUD_text.setFillColor(MONOCROME_TEXT_GREEN);
525
526     this->render_window_ptr->draw(HUD_text);
527
528
529     // 2. second line (top)
530     HUD_string = "CUMULATIVE EMISSIONS: ";
```

```

531 HUD_string += std::to_string(this->cumulative_emissions_tonnes);
532 HUD_string += " tonnes (CO2e)";
533
534 HUD_string += " LIFETIME LIMIT: ";
535 HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
536 HUD_string += " tonnes (CO2e)";
537
538 HUD_text.setString(HUD_string);
539
540 HUD_text.setPosition(
541     (800 - HUD_text.getLocalBounds().width) / 2,
542     35
543 );
544
545 this->render_window_ptr->draw(HUD_text);
546
547
548 // 3. third line (bottom)
549 HUD_string = "GAME PHASE: ";
550
551 switch (this->game_phase) {
552     case (GamePhase :: BUILD_SETTLEMENT): {
553         HUD_string += "BUILD SETTLEMENT";
554
555         break;
556     }
557
558     case (GamePhase :: SYSTEM_MANAGEMENT): {
559         HUD_string += "SYSTEM MANAGEMENT";
560
561         break;
562     }
563
564     case (GamePhase :: LOSS_EMISSIONS): {
565         HUD_string += "LOSS (EMISSIONS)";
566
567         break;
568     }
569
570     case (GamePhase :: LOSS_DEMAND): {
571         HUD_string += "LOSS (DEMAND)";
572
573         break;
574     }
575
576     case (GamePhase :: LOSS_CREDITS): {
577         HUD_string += "LOSS (CREDITS)";
578
579         break;
580     }
581
582     case (GamePhase :: VICTORY): {
583         HUD_string += "VICTORY";
584
585         break;
586     }
587
588     default: {
589         HUD_string += "???";
590
591         break;
592     }
593 }
594
595 HUD_string += " TURN: ";
596 HUD_string += std::to_string(this->turn);
597
598 HUD_text.setString(HUD_string);
599
600 HUD_text.setPosition(
601     (800 - HUD_text.getLocalBounds().width) / 2,
602     GAME_HEIGHT - 35
603 );
604
605 this->render_window_ptr->draw(HUD_text);
606
607 return;
608 }
609
610 /* __drawHUD() */

```

4.5.3.4 __handleKeyPressEvents()

```
void Game::__handleKeyPressEvents (
    void ) [private]
```

Helper method to handle key press events.

```
59 {
60     switch (this->event.key.code) {
61         case (sf::Keyboard::Tilde): {
62             this->__toggleFrameClockOverlay();
63
64             break;
65         }
66
67
68         case (sf::Keyboard::Tab): {
69             this->hex_map_ptr->toggleResourceOverlay();
70
71             break;
72         }
73
74
75         default: {
76             // do nothing!
77
78             break;
79         }
80     }
81
82     return;
83 } /* __handleKeyPressEvents() */
```

4.5.3.5 __handleMouseButtonEvents()

```
void Game::__handleMouseButtonEvents (
    void ) [private]
```

Helper method to handle mouse button events.

```
98 {
99     switch (this->event.mouseButton.button) {
100         case (sf::Mouse::Left): {
101             //...
102
103             break;
104         }
105
106
107         case (sf::Mouse::Right): {
108             //...
109
110             break;
111         }
112
113
114         default: {
115             // do nothing!
116
117             break;
118         }
119     }
120
121     return;
122 } /* __handleMouseButtonEvents() */
```

4.5.3.6 __insufficientCreditsAlarm()

```
void Game::__insufficientCreditsAlarm (
    void ) [private]
```

Helper method to sound and display and insufficient credits alarm.

```
354 {
355     // 1. sound buzzer
356     this->assets_manager_ptr->getSound("insufficient credits")->play();
357
358     // 2. construct alarm text and backing rectangle
359     sf::Text insufficient_credits_text(
360         "INSUFFICIENT CREDITS",
361         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
362         32
363     );
364
365     insufficient_credits_text.setOrigin(
366         insufficient_credits_text.getLocalBounds().width / 2,
367         insufficient_credits_text.getLocalBounds().height / 2
368     );
369
370     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
371
372     sf::RectangleShape backing_rectangle(
373         sf::Vector2f(
374             1.1 * insufficient_credits_text.getLocalBounds().width,
375             1.5 * insufficient_credits_text.getLocalBounds().height
376         )
377     );
378
379     backing_rectangle.setFill-color(RESOURCE_CHIP_GREY);
380
381     backing_rectangle.setOrigin(
382         backing_rectangle.getLocalBounds().width / 2,
383         backing_rectangle.getLocalBounds().height / 2
384     );
385
386     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
387
388     // 3. display loop (blocking ~3 seconds)
389     bool red_flag = true;
390     int alarm_frame = 0;
391     double time_since_alarm_s = 0;
392
393     sf::Clock alarm_clock;
394
395     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
396
397         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
398
399         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
400             while (this->render_window_ptr->pollEvent(this->event)) {
401                 // do nothing!
402             }
403
404             this->render_window_ptr->clear();
405
406             this->hex_map_ptr->draw();
407             this->context_menu_ptr->draw();
408             this->__draw();
409
410             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
411                 if (red_flag) {
412                     red_flag = false;
413                 }
414
415                 else {
416                     red_flag = true;
417                 }
418             }
419
420             if (red_flag) {
421                 insufficient_credits_text.setFill-color(MONOCHROME_TEXT_RED);
422             }
423
424             else {
425                 insufficient_credits_text.setFill-color(sf::Color(255, 255, 255));
426             }
427
428             this->render_window_ptr->draw(backing_rectangle);
429             this->render_window_ptr->draw(insufficient_credits_text);
430
431             alarm_frame++;
432         }
433     }
```



```

432         this->render_window_ptr->display();
433
434         alarm_frame++;
435         this->frame++;
436     }
437 }
438
439 return;
440 } /* __insufficientCreditsAlarm( */

```

4.5.3.7 __processEvent()

```

void Game::__processEvent (
    void ) [private]

```

Helper method to process [Game](#). To be called once per event.

```

138 {
139     if (this->event.type == sf::Event::Closed) {
140         this->quit_game = true;
141         this->game_loop_broken = true;
142     }
143
144     if (this->event.type == sf::Event::KeyPressed) {
145         this->__handleKeyPressEvents();
146     }
147
148     if (this->event.type == sf::Event::MouseButtonPressed) {
149         this->__handleMouseButtonEvents();
150     }
151
152     return;
153 } /* __processEvent() */

```

4.5.3.8 __processMessage()

```

void Game::__processMessage (
    void ) [private]

```

Helper method to process [Game](#). To be called once per message.

```

251 {
252     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
253         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
254
255         if (game_channel_message.subject == "quit game") {
256             this->quit_game = true;
257             this->game_loop_broken = true;
258
259             std::cout << "Quit game message received by " << this << std::endl;
260             this->message_hub.popMessage(GAME_CHANNEL);
261         }
262
263         if (game_channel_message.subject == "restart game") {
264             this->game_loop_broken = true;
265
266             std::cout << "Restart game message received by " << this << std::endl;
267             this->message_hub.popMessage(GAME_CHANNEL);
268         }
269
270         if (game_channel_message.subject == "state request") {
271             std::cout << "Game state request message received by " << this << std::endl;
272
273             this->__sendGameStateMessage();
274             this->message_hub.popMessage(GAME_CHANNEL);
275         }
276
277         if (game_channel_message.subject == "credits spent") {
278             this->credits -= game_channel_message.int_payload["credits spent"];
279
280             std::cout << "Credits spent message (" <<

```

```

281         game_channel_message.int_payload["credits spent"] << " ) received by "
282         << this << std::endl;
283
284         std::cout << "Current credits (Game): " << this->credits << " K" <<
285         std::endl;
286
287         this->message_hub.popMessage(GAME_CHANNEL);
288     }
289
290     if (game_channel_message.subject == "insufficient credits") {
291         std::cout << "Insufficient credits message received by " << this <<
292         std::endl;
293
294         this->__insufficientCreditsAlarm();
295
296         this->message_hub.popMessage(GAME_CHANNEL);
297     }
298
299     if (game_channel_message.subject == "update game phase") {
300         std::cout << "Update game phase message received by " << this << std::endl;
301
302         if (
303             game_channel_message.string_payload["game phase"] == "system management"
304         ) {
305             this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
306             this->population = STARTING_POPULATION;
307             this->turn++;
308         }
309
310         else if (
311             game_channel_message.string_payload["game phase"] == "loss emissions"
312         ) {
313             this->game_phase = GamePhase :: LOSS_EMISSIONS;
314         }
315
316         else if (
317             game_channel_message.string_payload["game phase"] == "loss demand"
318         ) {
319             this->game_phase = GamePhase :: LOSS_DEMAND;
320         }
321
322         else if (
323             game_channel_message.string_payload["game phase"] == "loss credits"
324         ) {
325             this->game_phase = GamePhase :: LOSS_CREDITS;
326         }
327
328         else if (
329             game_channel_message.string_payload["game phase"] == "victory"
330         ) {
331             this->game_phase = GamePhase :: VICTORY;
332         }
333
334         this->message_hub.popMessage(GAME_CHANNEL);
335     }
336 }
337
338 return;
339 } /* __processMessage() */

```

4.5.3.9 __sendGameStateMessage()

```

void Game::__sendGameStateMessage (
    void ) [private]

```

Helper method to format and send a game state message.

```

168 {
169     Message game_state_message;
170
171     game_state_message.channel = GAME_STATE_CHANNEL;
172     game_state_message.subject = "game state";
173
174     game_state_message.int_payload["year"] = this->year;
175     game_state_message.int_payload["month"] = this->month;
176     game_state_message.int_payload["population"] = this->population;
177     game_state_message.int_payload["credits"] = this->credits;
178     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
179     game_state_message.int_payload["cumulative_emissions_tonnes"] =

```

```

180         this->cumulative_emissions_tonnes;
181
182     switch (this->game_phase) {
183     case (GamePhase :: BUILD_SETTLEMENT): {
184         game_state_message.string_payload["game phase"] = "build settlement";
185
186         break;
187     }
188
189     case (GamePhase :: SYSTEM_MANAGEMENT): {
190         game_state_message.string_payload["game phase"] = "system management";
191
192         break;
193     }
194
195     case (GamePhase :: LOSS_EMISSIONS): {
196         game_state_message.string_payload["game phase"] = "loss emissions";
197
198         break;
199     }
200
201     case (GamePhase :: LOSS_DEMAND): {
202         game_state_message.string_payload["game phase"] = "loss demand";
203
204         break;
205     }
206
207     case (GamePhase :: LOSS_CREDITS): {
208         game_state_message.string_payload["game phase"] = "loss credits";
209
210         break;
211     }
212
213     case (GamePhase :: VICTORY): {
214         game_state_message.string_payload["game phase"] = "victory";
215
216         break;
217     }
218
219     default: {
220         // do nothing!
221
222         break;
223     }
224 }
225
226 this->message_hub.sendMessage(game_state_message);
227
228 std::cout << "Game state message sent by " << this << std::endl;
229 return;
230 }
231 /* __sendGameStateMessage() */

```

4.5.3.10 __toggleFrameClockOverlay()

```

void Game::__toggleFrameClockOverlay (
    void ) [private]

```

Helper method to toggle frame clock overlay.

```

34 {
35     if (this->show_frame_clock_overlay) {
36         this->show_frame_clock_overlay = false;
37     }
38
39     else {
40         this->show_frame_clock_overlay = true;
41     }
42
43     return;
44 }
45 /* __toggleFrameClockOverlay() */

```

4.5.3.11 run()

```
bool Game::run (
    void )
```

Method to run game (defines game loop).

Returns

Boolean indicating whether to quit (true) or create a new [Game](#) instance (false).

```
731 {
732     // 1. play brand animation
733     //...
734
735     // 2. show splash screen
736     //...
737
738     // 3. start game loop
739     while (not this->game_loop_broken) {
740         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
741
742         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
743             // 6.1. process events
744             while (this->render_window_ptr->pollEvent(this->event)) {
745                 this->hex_map_ptr->processEvent();
746                 this->context_menu_ptr->processEvent();
747                 this->__processEvent();
748             }
749
750             // 6.2. process messages
751             while (this->message_hub.hasTraffic()) {
752                 this->hex_map_ptr->processMessage();
753                 this->context_menu_ptr->processMessage();
754                 this->__processMessage();
755             }
756
757             // 6.3. draw frame
758             this->render_window_ptr->clear();
759
760             this->hex_map_ptr->draw();
761             this->context_menu_ptr->draw();
762             this->__draw();
763
764             this->render_window_ptr->display();
765
766             // 6.4. increment frame
767             this->frame++;
768         }
769     }
770
771     return this->quit_game;
772 }
773
774 /* run() */
```

4.5.4 Member Data Documentation

4.5.4.1 assets_manager_ptr

```
AssetsManager* Game::assets_manager_ptr [private]
```

A pointer to the assets manager.

4.5.4.2 clock

```
sf::Clock Game::clock
```

The game clock.

4.5.4.3 context_menu_ptr

```
ContextMenu* Game::context_menu_ptr
```

Pointer to the context menu.

4.5.4.4 credits

```
int Game::credits
```

Current balance of credits.

4.5.4.5 cumulative_emissions_tonnes

```
int Game::cumulative_emissions_tonnes
```

Cumulative emissions [tonnes] (1 tonne = 1000 kg).

4.5.4.6 demand_MWh

```
int Game::demand_MWh
```

Current energy demand [MWh].

4.5.4.7 event

```
sf::Event Game::event
```

The game events class.

4.5.4.8 frame

```
unsigned long long int Game::frame
```

The current frame of the game.

4.5.4.9 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.10 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.11 hex_map_ptr

```
HexMap* Game::hex_map_ptr
```

Pointer to the hex map (defines game world).

4.5.4.12 message_hub

```
MessageHub Game::message_hub
```

The message hub (for inter-object message traffic).

4.5.4.13 month

```
int Game::month
```

Current game month.

4.5.4.14 population

```
int Game::population
```

Current population.

4.5.4.15 quit_game

```
bool Game::quit_game
```

Boolean indicating whether to quit (true) or create a new [Game](#) instance (false).

4.5.4.16 render_window_ptr

```
sf::RenderWindow* Game::render_window_ptr [private]
```

A pointer to the render window.

4.5.4.17 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

Boolean indicating whether or not to show frame and clock overlay.

4.5.4.18 time_since_start_s

```
double Game::time_since_start_s
```

The time elapsed [s] since the start of the game.

4.5.4.19 turn

```
int Game::turn = 0
```

The current game turn.

4.5.4.20 year

```
int Game::year
```

Current game year.

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

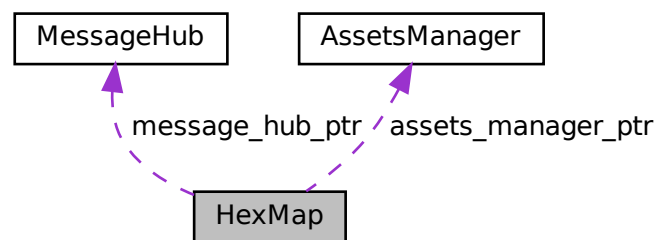
- header/[Game.h](#)
- source/[Game.cpp](#)

4.6 HexMap Class Reference

A class which defines a hex map of hex tiles.

```
#include <HexMap.h>
```

Collaboration diagram for HexMap:



Public Member Functions

- [HexMap](#) (int, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor (intended) for the [HexMap](#) class.
- void [assess](#) (void)
Method to assess the resource of the selected tile.
- void [reroll](#) (void)
Method to re-roll the hex map.
- void [toggleResourceOverlay](#) (void)
Method to toggle the hex map resource overlay.
- void [processEvent](#) (void)
Method to process [HexMap](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [HexMap](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex map to the render window. To be called once per frame.
- void [clear](#) (void)
Method to clear the hex map.
- [~HexMap](#) (void)
Destructor for the [HexMap](#) class.

Public Attributes

- bool [show_resource](#)
A boolean which indicates whether or not to show resource value.
- bool [tile_selected](#)
A boolean which indicates if a tile is currently selected.
- int [n_layers](#)
The number of layers in the hex map.
- int [n_tiles](#)
The number of tiles in the hex map.
- int [frame](#)
The current frame of this object.
- double [position_x](#)
The x position of the hex map's origin (i.e. central) tile.
- double [position_y](#)
The y position of the hex map's origin (i.e. central) tile.
- sf::RectangleShape [glass_screen](#)
To give the effect of an old glass screen over the hex map.
- std::vector< double > [tile_position_x_vec](#)
A vector of tile x positions.
- std::vector< double > [tile_position_y_vec](#)
A vector of tile y position.
- std::vector< [HexTile](#) * > [border_tiles_vec](#)
A vector of pointers to the border tiles.
- std::map< double, std::map< double, [HexTile](#) * > > [hex_map](#)
A position-indexed, nested map of hex tiles.
- std::vector< [HexTile](#) * > [hex_draw_order_vec](#)
A vector of hex tiles, in drawing order.

Private Member Functions

- void [__setUpGlassScreen](#) (void)
Helper method to set up glass screen effect (drawable).
- void [__layTiles](#) (void)
Helper method to lay the hex tiles down to generate the game world.
- void [__buildDrawOrderVector](#) (void)
Helper method to build tile drawing order vector.
- std::vector< double > [__getNoise](#) (int, int=128)
Helper method to generate a vector of noise, with values mapped to the closed interval [0, 1]. Applies a random cosine series approach.
- void [__procedurallyGenerateTileTypes](#) (void)
Helper method to procedurally generate tile types and set tiles accordingly.
- std::vector< double > [__getValidMapIndexPositions](#) (double, double)
Helper method to translate given position into valid index position for a.
- std::vector< [HexTile](#) * > [__getNeighboursVector](#) ([HexTile](#) *)
Helper method to assemble a vector pointers to all neighbours of the given tile.
- [TileType](#) [__getMajorityTileType](#) ([HexTile](#) *)
Function to return majority tile type of a tile and its neighbours. If no clear majority, simply returns the type of the given tile.
- void [__smoothTileTypes](#) (void)
Helper method to smooth tile types using a majority rules approach.

- bool [__isLakeTouchingOcean](#) (HexTile *)
- void [__enforceOceanContinuity](#) (void)

Helper method to scan tiles and enforce ocean continuity. That is to say, if a lake tile is found to be in contact with an ocean tile, then it becomes ocean.
- void [__procedurallyGenerateTileResources](#) (void)

Helper method to procedurally generate tile resources and set tiles accordingly.
- void [__assembleHexMap](#) (void)

Helper method to assemble the hex map.
- HexTile * [__getSelectedTile](#) (void)

Helper method to get pointer to selected tile.
- void [__handleKeyPressEvents](#) (void)

Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)

Helper method to handle mouse button events.
- void [__sendNoTileSelectedMessage](#) (void)

Helper method to format and send message on no tile selected.
- void [__assessNeighbours](#) (HexTile *)

Helper method to assess all neighbours of the given tile.

Private Attributes

- sf::Event * [event_ptr](#)

A pointer to the event class.
- sf::RenderWindow * [render_window_ptr](#)

A pointer to the render window.
- AssetsManager * [assets_manager_ptr](#)

A pointer to the assets manager.
- MessageHub * [message_hub_ptr](#)

A pointer to the message hub.

4.6.1 Detailed Description

A class which defines a hex map of hex tiles.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 HexMap()

```
HexMap::HexMap (
    int n_layers,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor (intended) for the [HexMap](#) class.

Parameters

<i>n_layers</i>	The number of layers in the HexMap .
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```

1082 {
1083     // 1. set attributes
1084
1085     // 1.1. private
1086     this->event_ptr = event_ptr;
1087     this->render_window_ptr = render_window_ptr;
1088
1089     this->assets_manager_ptr = assets_manager_ptr;
1090     this->message_hub_ptr = message_hub_ptr;
1091
1092     // 1.2. public
1093     this->show_resource = false;
1094     this->tile_selected = false;
1095
1096     this->frame = 0;
1097
1098     this->n_layers = n_layers;
1099     if (this->n_layers < 0) {
1100         this->n_layers = 0;
1101     }
1102
1103     this->position_x = 400;
1104     this->position_y = 400;
1105
1106     // 2. assemble n layer hex map
1107     this->__assembleHexMap();
1108
1109     // 3. set up and position drawable attributes
1110     this->__setUpGlassScreen();
1111
1112     // 4. add message channel(s)
1113     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1114     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1115     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1116     this->message_hub_ptr->addChannel(HEX_MAP_CHANNEL);
1117
1118     std::cout << "HexMap constructed at " << this << std::endl;
1119
1120     return;
1121 } /* HexMap(), intended */

```

4.6.2.2 ~HexMap()

```

HexMap::~HexMap (
    void )

```

Destructor for the [HexMap](#) class.

```

1413 {
1414     this->clear();
1415
1416     std::cout << "HexMap at " << this << " destroyed" << std::endl;
1417
1418     return;
1419 } /* ~HexMap() */

```

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

```
void HexMap::__assembleHexMap (
    void ) [private]
```

Helper method to assemble the hex map.

```
841 {
842     // 1. seed RNG (using milliseconds since 1 Jan 1970)
843     unsigned long long int milliseconds_since_epoch =
844         std::chrono::duration_cast<std::chrono::milliseconds>(
845             std::chrono::system_clock::now().time_since_epoch()
846         ).count();
847     srand(milliseconds_since_epoch);
848
849     // 2. lay tiles
850     this->__layTiles();
851     this->__buildDrawOrderVector();
852
853     // 3. procedurally generate types
854     this->__procedurallyGenerateTileTypes();
855
856     // 4. procedurally generate resources
857     this->__procedurallyGenerateTileResources();
858
859     return;
860 } /* __assembleHexMap() */
```

4.6.3.2 __assessNeighbours()

```
void HexMap::__assessNeighbours (
    HexTile * hex_ptr ) [private]
```

Helper method to assess all neighbours of the given tile.

Parameters

<i>Pointer</i>	to the tile whose neighbours are to be assessed.
----------------	--

```
1033 {
1034     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
1035
1036     for (size_t i = 0; i < neighbours_vec.size(); i++) {
1037         neighbours_vec[i]->assess();
1038     }
1039
1040     return;
1041 } /* __assessNeighbours() */
```

4.6.3.3 __buildDrawOrderVector()

```
void HexMap::__buildDrawOrderVector (
    void ) [private]
```

Helper method to build tile drawing order vector.

```
239 {
240     // 1. build temp list of tiles
241     std::list<HexTile*> temp_list;
242
243     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
244     std::map<double, HexTile*>::iterator hex_map_iter_y;
245     for (
246         hex_map_iter_x = this->hex_map.begin();
```

```

247     hex_map_iter_x != this->hex_map.end();
248     hex_map_iter_x++
249 ) {
250     for (
251         hex_map_iter_y = hex_map_iter_x->second.begin();
252         hex_map_iter_y != hex_map_iter_x->second.end();
253         hex_map_iter_y++
254     ) {
255         temp_list.push_back(hex_map_iter_y->second);
256     }
257 }
258
259 // 2. move elements from temp list to drawing order vector
260 double min_position_y = 0;
261 std::list<HexTile*>::iterator list_iter;
262
263 while (not temp_list.empty()) {
264     // 2.1. determine min y position
265     min_position_y = std::numeric_limits<double>::infinity();
266
267     for (
268         list_iter = temp_list.begin();
269         list_iter != temp_list.end();
270         list_iter++
271     ) {
272         if ((*list_iter)->position_y < min_position_y) {
273             min_position_y = (*list_iter)->position_y;
274         }
275     }
276
277     // 2.2 move min y list elements to drawing order vec
278     list_iter = temp_list.begin();
279     while (list_iter != temp_list.end()) {
280         if ((*list_iter)->position_y == min_position_y) {
281             this->hex_draw_order_vec.push_back((*list_iter));
282             list_iter = temp_list.erase(list_iter);
283         }
284         else {
285             list_iter++;
286         }
287     }
288 }
289 }
290
291 return;
292 } /* __buildDrawOrderVector() */

```

4.6.3.4 __enforceOceanContinuity()

```

void HexMap::__enforceOceanContinuity (
    void ) [private]

```

Helper method to scan tiles and enforce ocean continuity. That is to say, if a lake tile is found to be in contact with an ocean tile, then it becomes ocean.

```

752 {
753     std::cout << "enforcing ocean continuity ..." << std::endl;
754
755     bool tile_changed = false;
756
757     // 1. scan tiles and enforce (where appropriate)
758     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
759     std::map<double, HexTile*>::iterator hex_map_iter_y;
760     HexTile* hex_ptr;
761     for (
762         hex_map_iter_x = this->hex_map.begin();
763         hex_map_iter_x != this->hex_map.end();
764         hex_map_iter_x++
765     ) {
766         for (
767             hex_map_iter_y = hex_map_iter_x->second.begin();
768             hex_map_iter_y != hex_map_iter_x->second.end();
769             hex_map_iter_y++
770         ) {
771             hex_ptr = hex_map_iter_y->second;
772
773             if (this->__isLakeTouchingOcean(hex_ptr)) {
774                 hex_ptr->setTileType(TileType :: OCEAN);
775                 tile_changed = true;

```

```

776         }
777     }
778 }
779
780 if (tile_changed) {
781     this->__enforceOceanContinuity();
782 }
783 else {
784     return;
785 }
786 } /* __enforceOceanContinuity() */

```

4.6.3.5 __getMajorityTileType()

```

TileType HexMap::__getMajorityTileType (
    HexTile * hex_ptr ) [private]

```

Function to return majority tile type of a tile and its neighbours. If no clear majority, simply returns the type of the given tile.

Parameters

<i>hex_ptr</i>	Pointer to the given tile.
----------------	----------------------------

Returns

The majority tile type of the tile and its neighbours. If no clear majority type, then the type of the given tile is simply returned.

```

608 {
609     // 1. init type count map
610     std::map<TileType, int> type_count_map;
611     type_count_map[hex_ptr->tile_type] = 1;
612
613     // 2. survey neighbours, count type instances
614     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
615
616     for (size_t i = 0; i < neighbours_vec.size(); i++) {
617         if (type_count_map.count(neighbours_vec[i]->tile_type) <= 0) {
618             type_count_map[neighbours_vec[i]->tile_type] = 1;
619         }
620         else {
621             type_count_map[neighbours_vec[i]->tile_type] += 1;
622         }
623     }
624
625     // 3. find majority tile type
626     int max_count = -1 * std::numeric_limits<int>::infinity();
627     TileType majority_tile_type = hex_ptr->tile_type;
628
629     std::map<TileType, int>::iterator map_iter;
630     for (
631         map_iter = type_count_map.begin();
632         map_iter != type_count_map.end();
633         map_iter++)
634     ){
635         if (map_iter->second > max_count) {
636             max_count = map_iter->second;
637             majority_tile_type = map_iter->first;
638         }
639     }
640
641     // 4. detect ties
642     for (
643         map_iter = type_count_map.begin();
644         map_iter != type_count_map.end();
645         map_iter++)
646     ){
647         if (
648             map_iter->second == max_count and
649             map_iter->first != majority_tile_type

```

```

650         } {
651             majority_tile_type = hex_ptr->tile_type;
652             break;
653         }
654     }
655
656     return majority_tile_type;
657 } /* __getMajorityTileType() */

```

4.6.3.6 __getNeighboursVector()

```

std::vector< HexTile * > HexMap::__getNeighboursVector (
    HexTile * hex_ptr ) [private]

```

Helper method to assemble a vector pointers to all neighbours of the given tile.

Parameters

<i>hex_ptr</i>	A pointer to the given tile.
----------------	------------------------------

Returns

A vector of pointers to all neighbours of the given tile.

```

550 {
551     std::vector<HexTile*> neighbours_vec;
552
553     // 1. build potential neighbour positions
554     std::vector<double> potential_neighbour_x_vec(6, 0);
555     std::vector<double> potential_neighbour_y_vec(6, 0);
556
557     for (int i = 0; i < 6; i++) {
558         potential_neighbour_x_vec[i] = hex_ptr->position_x +
559             2 * hex_ptr->minor_radius * cos((60 * i) * (M_PI / 180));
560
561         potential_neighbour_y_vec[i] = hex_ptr->position_y +
562             2 * hex_ptr->minor_radius * sin((60 * i) * (M_PI / 180));
563     }
564
565     // 2. populate neighbours vector
566     std::vector<double> map_index_positions;
567     double potential_x = 0;
568     double potential_y = 0;
569
570     for (int i = 0; i < 6; i++) {
571         potential_x = potential_neighbour_x_vec[i];
572         potential_y = potential_neighbour_y_vec[i];
573
574         map_index_positions = this->__getValidMapIndexPositions(
575             potential_x,
576             potential_y
577         );
578
579         if (not (map_index_positions[0] == -1)) {
580             neighbours_vec.push_back(
581                 this->hex_map[map_index_positions[0]][map_index_positions[1]]
582             );
583         }
584     }
585
586     return neighbours_vec;
587 } /* __getNeighbourVector() */

```

4.6.3.7 __getNoise()

```

std::vector< double > HexMap::__getNoise (
    int n_elements,
    int n_components = 128 ) [private]

```

Helper method to generate a vector of noise, with values mapped to the closed interval [0, 1]. Applies a random cosine series approach.

Parameters

<i>n_elements</i>	The number of elements in the generated noise vector.
<i>n_components</i>	The number of components to use in the random cosine series. Defaults to 64.

Returns

A vector of noise, with values mapped to the closed interval [0, 1].

```

315 {
316     // 1. generate random amplitude, wave number, direction, and phase vectors
317     std::vector<double> random_amplitude_vec(n_components, 0);
318     std::vector<double> random_wave_number_vec(n_components, 0);
319     std::vector<double> random_frequency_vec(n_components, 0);
320     std::vector<double> random_direction_vec(n_components, 0);
321     std::vector<double> random_phase_vec(n_components, 0);
322
323     for (int i = 0; i < n_components; i++) {
324         random_amplitude_vec[i] = 10 * ((double)rand() / RAND_MAX);
325
326         random_wave_number_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
327
328         random_frequency_vec[i] = ((double)rand() / RAND_MAX);
329
330         random_direction_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
331
332         random_phase_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
333     }
334
335     // 2. generate noise vec
336     double amp = 0;
337     double wave_no = 0;
338     double freq = 0;
339     double dir = 0;
340     double phase = 0;
341
342     double x = 0;
343     double y = 0;
344     double t = time(NULL);
345
346     double max_noise = -1 * std::numeric_limits<double>::infinity();
347     double min_noise = std::numeric_limits<double>::infinity();
348
349     double noise = 0;
350     std::vector<double> noise_vec(n_elements, 0);
351
352     for (int i = 0; i < n_elements; i++) {
353         x = this->tile_position_x_vec[i] - this->position_x;
354         y = this->tile_position_y_vec[i] - this->position_y;
355
356         for (int j = 0; j < n_components; j++) {
357             amp = random_amplitude_vec[j];
358             wave_no = random_wave_number_vec[j];
359             freq = random_frequency_vec[j];
360             dir = random_direction_vec[j];
361             phase = random_phase_vec[j];
362
363             noise += (amp / (j + 1)) * cos(
364                 wave_no * (j + 1) * (x * sin(dir) + y * cos(dir)) +
365                 2 * M_PI * (j + 1) * freq * t +
366                 phase
367             );
368         }
369
370         noise_vec[i] = noise;
371
372         if (noise > max_noise) {
373             max_noise = noise;
374         }
375
376         else if (noise < min_noise) {
377             min_noise = noise;
378         }
379
380         noise = 0;
381     }
382

```



```

383 // 3. normalize noise vec
384 for (int i = 0; i < n_elements; i++) {
385     noise_vec[i] = (noise_vec[i] - min_noise) / (max_noise - min_noise);
386
387     if (noise_vec[i] < 0) {
388         noise_vec[i] = 0;
389     }
390     else if (noise_vec[i] > 1) {
391         noise_vec[i] = 1;
392     }
393 }
394
395 return noise_vec;
396 } /* __getNoise() */

```

4.6.3.8 __getSelectedTile()

```

HexTile * HexMap::__getSelectedTile (
    void ) [private]

```

Helper method to get pointer to selected tile.

Returns

Pointer to selected tile (or NULL if no tile selected).

```

877 {
878     HexTile* selected_tile_ptr = NULL;
879
880     bool break_flag = false;
881     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
882     std::map<double, HexTile*>::iterator hex_map_iter_y;
883
884     for (
885         hex_map_iter_x = this->hex_map.begin();
886         hex_map_iter_x != this->hex_map.end();
887         hex_map_iter_x++
888     ) {
889         for (
890             hex_map_iter_y = hex_map_iter_x->second.begin();
891             hex_map_iter_y != hex_map_iter_x->second.end();
892             hex_map_iter_y++
893         ) {
894             if (hex_map_iter_y->second->is_selected) {
895                 selected_tile_ptr = hex_map_iter_y->second;
896                 break_flag = true;
897             }
898
899             if (break_flag) {
900                 break;
901             }
902         }
903
904         if (break_flag) {
905             break;
906         }
907     }
908
909     return selected_tile_ptr;
910 } /* __getSelectedTile() */

```

4.6.3.9 __getValidMapIndexPositions()

```

std::vector< double > HexMap::__getValidMapIndexPositions (
    double potential_x,
    double potential_y ) [private]

```

Helper method to translate given position into valid index position for a.

Parameters

<i>potential</i> ↔ _x	The potential x position of the tile.
<i>potential</i> ↔ _y	The potential y position of the tile.

Returns

A vector of positions, either valid for indexing into the hex map, or sentinel values (-1) if invalid.

```

496 {
497     std::vector<double> map_index_positions = {-1, -1};
498
499     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
500     std::map<double, HexTile*>::iterator hex_map_iter_y;
501     HexTile* hex_ptr;
502
503     double distance = 0;
504
505     for (
506         hex_map_iter_x = this->hex_map.begin();
507         hex_map_iter_x != this->hex_map.end();
508         hex_map_iter_x++
509     ) {
510         for (
511             hex_map_iter_y = hex_map_iter_x->second.begin();
512             hex_map_iter_y != hex_map_iter_x->second.end();
513             hex_map_iter_y++
514         ) {
515             hex_ptr = hex_map_iter_y->second;
516
517             distance = sqrt(
518                 pow(hex_ptr->position_x - potential_x, 2) +
519                 pow(hex_ptr->position_y - potential_y, 2)
520             );
521
522             if (distance <= hex_ptr->minor_radius / 4) {
523                 map_index_positions = {hex_ptr->position_x, hex_ptr->position_y};
524                 return map_index_positions;
525             }
526         }
527     }
528
529     return map_index_positions;
530 } /* __isInHexMap() */

```

4.6.3.10 __handleKeyPressEvents()

```

void HexMap::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

925 {
926     switch (this->event_ptr->key.code) {
927         case (sf::Keyboard::Escape): {
928             this->tile_selected = false;
929         }
930
931
932         default: {
933             // do nothing!
934
935             break;
936         }
937     }
938
939     return;
940 } /* __handleKeyPressEvents() */

```

4.6.3.11 __handleMouseButtonEvents()

```
void HexMap::__handleMouseButtonEvents (
    void ) [private]
```

Helper method to handle mouse button events.

```
955 {
956     switch (this->event_ptr->mouseButton.button) {
957         case (sf::Mouse::Left): {
958             HexTile* hex_ptr = this->__getSelectedTile();
959
960             if (hex_ptr != NULL) {
961                 this->tile_selected = true;
962             }
963
964             else if (this->tile_selected) {
965                 this->tile_selected = false;
966                 this->__sendNoTileSelectedMessage();
967             }
968
969             break;
970         }
971
972         case (sf::Mouse::Right): {
973             if (this->tile_selected) {
974                 this->tile_selected = false;
975                 this->__sendNoTileSelectedMessage();
976             }
977
978             break;
979         }
980
981         default: {
982             // do nothing!
983
984             break;
985         }
986     }
987
988     return;
989 }
990
991 } /* __handleMouseButtonEvents() */
```

4.6.3.12 __isLakeTouchingOcean()

```
bool HexMap::__isLakeTouchingOcean (
    HexTile * hex_ptr ) [private]

719 {
720     // 1. if not lake tile, return
721     if (not (hex_ptr->tile_type == TileType::LAKE)) {
722         return false;
723     }
724
725     // 2. scan neighbours for ocean tiles
726     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
727
728     for (size_t i = 0; i < neighbours_vec.size(); i++) {
729         if (neighbours_vec[i]->tile_type == TileType::OCEAN) {
730             return true;
731         }
732     }
733
734     return false;
735 } /* __isLakeTouchingOcean() */
```

4.6.3.13 __layTiles()

```
void HexMap::__layTiles (
    void ) [private]
```

Helper method to lay the hex tiles down to generate the game world.

```
54 {
55     this->n_tiles = 0;
56
57     // 1. add origin tile
58     HexTile* hex_ptr = new HexTile(
59         this->position_x,
60         this->position_y,
61         this->event_ptr,
62         this->render_window_ptr,
63         this->assets_manager_ptr,
64         this->message_hub_ptr
65     );
66
67     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
68     this->tile_position_x_vec.push_back(hex_ptr->position_x);
69     this->tile_position_y_vec.push_back(hex_ptr->position_y);
70     this->n_tiles++;
71
72
73     // 2. fill out first row (reflect across origin tile)
74     for (int i = 0; i < this->n_layers; i++) {
75         hex_ptr = new HexTile(
76             this->position_x + 2 * (i + 1) * hex_ptr->minor_radius,
77             this->position_y,
78             this->event_ptr,
79             this->render_window_ptr,
80             this->assets_manager_ptr,
81             this->message_hub_ptr
82         );
83
84         this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
85         this->tile_position_x_vec.push_back(hex_ptr->position_x);
86         this->tile_position_y_vec.push_back(hex_ptr->position_y);
87         this->n_tiles++;
88
89         if (i == this->n_layers - 1) {
90             this->border_tiles_vec.push_back(hex_ptr);
91         }
92
93         hex_ptr = new HexTile(
94             this->position_x - 2 * (i + 1) * hex_ptr->minor_radius,
95             this->position_y,
96             this->event_ptr,
97             this->render_window_ptr,
98             this->assets_manager_ptr,
99             this->message_hub_ptr
100     );
101
102     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
103     this->tile_position_x_vec.push_back(hex_ptr->position_x);
104     this->tile_position_y_vec.push_back(hex_ptr->position_y);
105     this->n_tiles++;
106
107     if (i == this->n_layers - 1) {
108         this->border_tiles_vec.push_back(hex_ptr);
109     }
110 }
111
112
113 // 3. fill out subsequent rows (reflect across first row)
114 HexTile* first_row_left_tile = hex_ptr;
115
116 int offset_count = 1;
117
118 double x_offset = 0;
119 double y_offset = 0;
120
121 for (
122     int row_width = 2 * this->n_layers;
123     row_width > this->n_layers;
124     row_width--
125 ) {
126     // 3.1. upper row
127     x_offset = first_row_left_tile->position_x +
128         2 * offset_count * first_row_left_tile->minor_radius *
129         cos(60 * (M_PI / 180));
130
131     y_offset = first_row_left_tile->position_y -
```

```

132         2 * offset_count * first_row_left_tile->minor_radius *
133         sin(60 * (M_PI / 180));
134
135     hex_ptr = new HexTile(
136         x_offset,
137         y_offset,
138         this->event_ptr,
139         this->render_window_ptr,
140         this->assets_manager_ptr,
141         this->message_hub_ptr
142     );
143
144     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
145     this->tile_position_x_vec.push_back(hex_ptr->position_x);
146     this->tile_position_y_vec.push_back(hex_ptr->position_y);
147     this->n_tiles++;
148
149     this->border_tiles_vec.push_back(hex_ptr);
150
151     for (int i = 1; i < row_width; i++) {
152         x_offset += 2 * first_row_left_tile->minor_radius;
153
154         hex_ptr = new HexTile(
155             x_offset,
156             y_offset,
157             this->event_ptr,
158             this->render_window_ptr,
159             this->assets_manager_ptr,
160             this->message_hub_ptr
161         );
162
163         this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
164         this->tile_position_x_vec.push_back(hex_ptr->position_x);
165         this->tile_position_y_vec.push_back(hex_ptr->position_y);
166         this->n_tiles++;
167
168         if (row_width == this->n_layers + 1 or i == row_width - 1) {
169             this->border_tiles_vec.push_back(hex_ptr);
170         }
171     }
172
173     // 3.2. lower row
174     x_offset = first_row_left_tile->position_x +
175         2 * offset_count * first_row_left_tile->minor_radius *
176         cos(60 * (M_PI / 180));
177
178     y_offset = first_row_left_tile->position_y +
179         2 * offset_count * first_row_left_tile->minor_radius *
180         sin(60 * (M_PI / 180));
181
182     hex_ptr = new HexTile(
183         x_offset,
184         y_offset,
185         this->event_ptr,
186         this->render_window_ptr,
187         this->assets_manager_ptr,
188         this->message_hub_ptr
189     );
190
191     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
192     this->tile_position_x_vec.push_back(hex_ptr->position_x);
193     this->tile_position_y_vec.push_back(hex_ptr->position_y);
194     this->n_tiles++;
195
196     this->border_tiles_vec.push_back(hex_ptr);
197
198     for (int i = 1; i < row_width; i++) {
199         x_offset += 2 * first_row_left_tile->minor_radius;
200
201         hex_ptr = new HexTile(
202             x_offset,
203             y_offset,
204             this->event_ptr,
205             this->render_window_ptr,
206             this->assets_manager_ptr,
207             this->message_hub_ptr
208         );
209
210         this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
211         this->tile_position_x_vec.push_back(hex_ptr->position_x);
212         this->tile_position_y_vec.push_back(hex_ptr->position_y);
213         this->n_tiles++;
214
215         if (row_width == this->n_layers + 1 or i == row_width - 1) {
216             this->border_tiles_vec.push_back(hex_ptr);
217         }
218     }

```

```

219
220     offset_count++;
221 }
222
223 return;
224 } /* __layTiles() */

```

4.6.3.14 __procedurallyGenerateTileResources()

```

void HexMap::__procedurallyGenerateTileResources (
    void ) [private]

```

Helper method to procedurally generate tile resources and set tiles accordingly.

```

801 {
802     // 1. get random cosine series noise vec
803     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
804
805     // 2. set tile resources based on random cosine series noise
806     int noise_idx = 0;
807
808     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
809     std::map<double, HexTile*>::iterator hex_map_iter_y;
810     for (
811         hex_map_iter_x = this->hex_map.begin();
812         hex_map_iter_x != this->hex_map.end();
813         hex_map_iter_x++
814     ) {
815         for (
816             hex_map_iter_y = hex_map_iter_x->second.begin();
817             hex_map_iter_y != hex_map_iter_x->second.end();
818             hex_map_iter_y++
819         ) {
820             hex_map_iter_y->second->setTileResource(noise_vec[noise_idx]);
821             noise_idx++;
822         }
823     }
824
825     return;
826 } /* __procedurallyGenerateTileResources() */

```

4.6.3.15 __procedurallyGenerateTileTypes()

```

void HexMap::__procedurallyGenerateTileTypes (
    void ) [private]

```

Helper method to procedurally generate tile types and set tiles accordingly.

```

411 {
412     // 1. get random cosine series noise vec
413     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
414
415     // 2. set initial tile types based on either random cosine series noise or white
416     //     noise (decided by coin toss)
417     int noise_idx = 0;
418
419     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
420     std::map<double, HexTile*>::iterator hex_map_iter_y;
421     for (
422         hex_map_iter_x = this->hex_map.begin();
423         hex_map_iter_x != this->hex_map.end();
424         hex_map_iter_x++
425     ) {
426         for (
427             hex_map_iter_y = hex_map_iter_x->second.begin();
428             hex_map_iter_y != hex_map_iter_x->second.end();
429             hex_map_iter_y++
430         ) {
431             if ((double)rand() / RAND_MAX > 0.5) {
432                 hex_map_iter_y->second->setTileType(noise_vec[noise_idx]);
433             }

```

```

434         else {
435             hex_map_iter_y->second->setTileType((double)rand() / RAND_MAX);
436         }
437         noise_idx++;
438     }
439 }
440
441 // 3. smooth tile types (majority rules)
442 this->__smoothTileTypes();
443
444 // 4. set border tile type to ocean
445 for (size_t i = 0; i < this->border_tiles_vec.size(); i++) {
446     this->border_tiles_vec[i]->setTileType(TileType :: OCEAN);
447 }
448
449 // 5. enforce ocean continuity (i.e. all lake tiles touching ocean become ocean)
450 this->__enforceOceanContinuity();
451
452 // 6. decorate tiles
453 for (
454     hex_map_iter_x = this->hex_map.begin();
455     hex_map_iter_x != this->hex_map.end();
456     hex_map_iter_x++
457 ) {
458     for (
459         hex_map_iter_y = hex_map_iter_x->second.begin();
460         hex_map_iter_y != hex_map_iter_x->second.end();
461         hex_map_iter_y++
462     ) {
463         hex_map_iter_y->second->decorateTile();
464     }
465 }
466
467 return;
468 } /* __procedurallyGenerateTileTypes() */

```

4.6.3.16 __sendNoTileSelectedMessage()

```

void HexMap::__sendNoTileSelectedMessage (
    void ) [private]

```

Helper method to format and send message on no tile selected.

```

1006 {
1007     Message no_tile_selected_message;
1008
1009     no_tile_selected_message.channel = NO_TILE_SELECTED_CHANNEL;
1010     no_tile_selected_message.subject = "no tile selected";
1011
1012     this->message_hub_ptr->sendMessage(no_tile_selected_message);
1013
1014     std::cout << "No tile selected message sent by " << this << std::endl;
1015     return;
1016 } /* __sendNoTileSelectedMessage() */

```

4.6.3.17 __setUpGlassScreen()

```

void HexMap::__setUpGlassScreen (
    void ) [private]

```

Helper method to set up glass screen effect (drawable).

```

34 {
35     this->glass_screen.setSize(sf::Vector2f(GAME_WIDTH, GAME_HEIGHT));
36     this->glass_screen.setFillColor(sf::Color(MONOCROME_SCREEN_BACKGROUND));
37
38     return;
39 } /* __setUpGlassScreen() */

```

4.6.3.18 __smoothTileTypes()

```
void HexMap::__smoothTileTypes (
    void ) [private]
```

Helper method to smooth tile types using a majority rules approach.

```
672 {
673     std::cout << "smoothing ..." << std::endl;
674
675     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
676     std::map<double, HexTile*>::iterator hex_map_iter_y;
677     HexTile* hex_ptr;
678     TileType majority_tile_type;
679
680     for (
681         hex_map_iter_x = this->hex_map.begin();
682         hex_map_iter_x != this->hex_map.end();
683         hex_map_iter_x++
684     ) {
685         for (
686             hex_map_iter_y = hex_map_iter_x->second.begin();
687             hex_map_iter_y != hex_map_iter_x->second.end();
688             hex_map_iter_y++
689         ) {
690             hex_ptr = hex_map_iter_y->second;
691             majority_tile_type = this->__getMajorityTileType(hex_ptr);
692
693             if (majority_tile_type != hex_ptr->tile_type) {
694                 hex_ptr->setTileType(majority_tile_type);
695             }
696         }
697     }
698
699     return;
700 } /* __smoothTileTypes() */
```

4.6.3.19 assess()

```
void HexMap::assess (
    void )
```

Method to assess the resource of the selected tile.

```
1136 {
1137     HexTile* selected_tile_ptr = this->__getSelectedTile();
1138     if (selected_tile_ptr != NULL) {
1139         selected_tile_ptr->assess();
1140     }
1141
1142     return;
1143 } /* assess() */
```

4.6.3.20 clear()

```
void HexMap::clear (
    void )
```

Method to clear the hex map.

```
1375 {
1376     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1377     std::map<double, HexTile*>::iterator hex_map_iter_y;
1378     for (
1379         hex_map_iter_x = this->hex_map.begin();
1380         hex_map_iter_x != this->hex_map.end();
1381         hex_map_iter_x++
1382     ) {
1383         for (
```



```

1384         hex_map_iter_y = hex_map_iter_x->second.begin();
1385         hex_map_iter_y != hex_map_iter_x->second.end();
1386         hex_map_iter_y++;
1387     } {
1388         delete hex_map_iter_y->second;
1389     }
1390 }
1391 this->hex_map.clear();
1392
1393 this->tile_position_x_vec.clear();
1394 this->tile_position_y_vec.clear();
1395 this->border_tiles_vec.clear();
1396
1397 return;
1398 } /* clear() */

```

4.6.3.21 draw()

```

void HexMap::draw (
    void )

```

Method to draw the hex map to the render window. To be called once per frame.

```

1314 {
1315     // 1. draw background
1316     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1317     glass_screen_colour.a = 255;
1318     this->glass_screen.setFillColor(glass_screen_colour);
1319
1320     this->render_window_ptr->draw(this->glass_screen);
1321
1322     // 2. draw tiles in drawing order
1323     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1324         this->hex_draw_order_vec[i]->draw();
1325     }
1326
1327     // 3. redraw selected tile
1328     HexTile* selected_tile_ptr = this->__getSelectedTile();
1329     if (selected_tile_ptr != NULL) {
1330         selected_tile_ptr->draw();
1331     }
1332
1333     // 4. draw resource overlay text indication
1334     if (this->show_resource) {
1335         sf::Text resource_overlay_text(
1336             "**** RENEWABLE RESOURCE OVERLAY ****",
1337             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1338             16
1339         );
1340
1341         resource_overlay_text.setPosition(
1342             (800 - resource_overlay_text.getLocalBounds().width) / 2,
1343             GAME_HEIGHT - 70
1344         );
1345
1346         resource_overlay_text.setFillColor(MONOCROME_TEXT_GREEN);
1347
1348         this->render_window_ptr->draw(resource_overlay_text);
1349     }
1350
1351     // 5. draw glass screen
1352     glass_screen_colour = this->glass_screen.getFillColor();
1353     glass_screen_colour.a = 40;
1354     this->glass_screen.setFillColor(glass_screen_colour);
1355
1356     this->render_window_ptr->draw(this->glass_screen);
1357
1358     this->frame++;
1359     return;
1360 } /* draw() */

```

4.6.3.22 processEvent()

```
void HexMap::processEvent (
    void )
```

Method to process [HexMap](#). To be called once per event.

```
1221 {
1222     // 1. process HexTile events
1223     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
1224     std::map<double, HexTile*>::iterator hex_map_iter_y;
1225     for (
1226         hex_map_iter_x = this->hex_map.begin();
1227         hex_map_iter_x != this->hex_map.end();
1228         hex_map_iter_x++
1229     ) {
1230         for (
1231             hex_map_iter_y = hex_map_iter_x->second.begin();
1232             hex_map_iter_y != hex_map_iter_x->second.end();
1233             hex_map_iter_y++
1234         ) {
1235             hex_map_iter_y->second->processEvent();
1236         }
1237     }
1238
1239     // 2. process HexMap events
1240     if (this->event_ptr->type == sf::Event::KeyPressed) {
1241         this->__handleKeyPressEvents();
1242     }
1243
1244     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1245         this->__handleMouseButtonEvents();
1246     }
1247
1248     return;
1249 } /* processEvent() */
```

4.6.3.23 processMessage()

```
void HexMap::processMessage (
    void )
```

Method to process [HexMap](#). To be called once per message.

```
1264 {
1265     // 1. process HexTile messages
1266     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
1267     std::map<double, HexTile*>::iterator hex_map_iter_y;
1268     for (
1269         hex_map_iter_x = this->hex_map.begin();
1270         hex_map_iter_x != this->hex_map.end();
1271         hex_map_iter_x++
1272     ) {
1273         for (
1274             hex_map_iter_y = hex_map_iter_x->second.begin();
1275             hex_map_iter_y != hex_map_iter_x->second.end();
1276             hex_map_iter_y++
1277         ) {
1278             hex_map_iter_y->second->processMessage();
1279         }
1280     }
1281
1282     // 2. process HexMap messages
1283     if (not this->message_hub_ptr->isEmpty(HEX_MAP_CHANNEL)) {
1284         Message hex_map_message = this->message_hub_ptr->receiveMessage(
1285             HEX_MAP_CHANNEL
1286         );
1287
1288         if (hex_map_message.subject == "assess neighbours") {
1289             HexTile* hex_ptr = this->__getSelectedTile();
1290             this->__assessNeighbours(hex_ptr);
1291
1292             std::cout << "Assess neighbours message received by " << this << std::endl;
1293             this->message_hub_ptr->popMessage(HEX_MAP_CHANNEL);
1294         }
1295     }
1296
1297     return;
1298 } /* processMessage() */
```

4.6.3.24 reroll()

```
void HexMap::reroll (
    void )
```

Method to re-roll the hex map.

```
1158 {
1159     this->clear();
1160     this->__assembleHexMap();
1161
1162     return;
1163 } /* reroll() */
```

4.6.3.25 toggleResourceOverlay()

```
void HexMap::toggleResourceOverlay (
    void )
```

Method to toggle the hex map resource overlay.

```
1178 {
1179     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1180     std::map<double, HexTile*>::iterator hex_map_iter_y;
1181     for (
1182         hex_map_iter_x = this->hex_map.begin();
1183         hex_map_iter_x != this->hex_map.end();
1184         hex_map_iter_x++
1185     ) {
1186         for (
1187             hex_map_iter_y = hex_map_iter_x->second.begin();
1188             hex_map_iter_y != hex_map_iter_x->second.end();
1189             hex_map_iter_y++
1190         ) {
1191             hex_map_iter_y->second->toggleResourceOverlay();
1192         }
1193     }
1194
1195     if (this->show_resource) {
1196         this->show_resource = false;
1197         this->assets_manager_ptr->getSound("resource overlay toggle off")->play();
1198     }
1199
1200     else {
1201         this->show_resource = true;
1202         this->assets_manager_ptr->getSound("resource overlay toggle on")->play();
1203     }
1204
1205     return;
1206 } /* toggleResourceOverlay() */
```

4.6.4 Member Data Documentation

4.6.4.1 assets_manager_ptr

```
AssetsManager* HexMap::assets_manager_ptr [private]
```

A pointer to the assets manager.

4.6.4.2 border_tiles_vec

```
std::vector<HexTile*> HexMap::border_tiles_vec
```

A vector of pointers to the border tiles.

4.6.4.3 event_ptr

```
sf::Event* HexMap::event_ptr [private]
```

A pointer to the event class.

4.6.4.4 frame

```
int HexMap::frame
```

The current frame of this object.

4.6.4.5 glass_screen

```
sf::RectangleShape HexMap::glass_screen
```

To give the effect of an old glass screen over the hex map.

4.6.4.6 hex_draw_order_vec

```
std::vector<HexTile*> HexMap::hex_draw_order_vec
```

A vector of hex tiles, in drawing order.

4.6.4.7 hex_map

```
std::map<double, std::map<double, HexTile*> > HexMap::hex_map
```

A position-indexed, nested map of hex tiles.

4.6.4.8 message_hub_ptr

```
MessageHub* HexMap::message_hub_ptr [private]
```

A pointer to the message hub.

4.6.4.9 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

4.6.4.10 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

4.6.4.11 position_x

```
double HexMap::position_x
```

The x position of the hex map's origin (i.e. central) tile.

4.6.4.12 position_y

```
double HexMap::position_y
```

The y position of the hex map's origin (i.e. central) tile.

4.6.4.13 render_window_ptr

```
sf::RenderWindow* HexMap::render_window_ptr [private]
```

A pointer to the render window.

4.6.4.14 show_resource

```
bool HexMap::show_resource
```

A boolean which indicates whether or not to show resource value.

4.6.4.15 tile_position_x_vec

```
std::vector<double> HexMap::tile_position_x_vec
```

A vector of tile x positions.

4.6.4.16 tile_position_y_vec

```
std::vector<double> HexMap::tile_position_y_vec
```

A vector of tile y position.

4.6.4.17 tile_selected

```
bool HexMap::tile_selected
```

A boolean which indicates if a tile is currently selected.

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

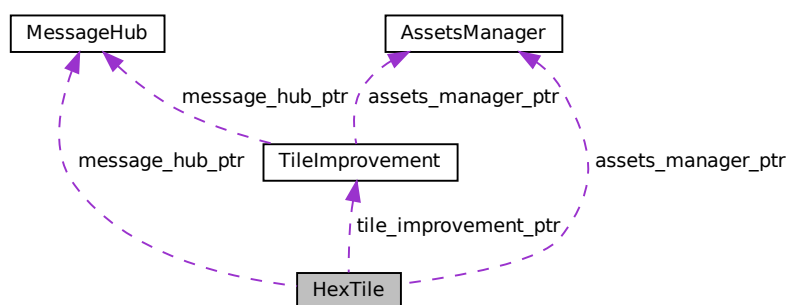
- header/[HexMap.h](#)
- source/[HexMap.cpp](#)

4.7 HexTile Class Reference

A class which defines a hex tile of the hex map.

```
#include <HexTile.h>
```

Collaboration diagram for HexTile:



Public Member Functions

- [HexTile](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [HexTile](#) class.
- void [setTileType](#) ([TileType](#))
Method to set the tile type (by enum value).
- void [setTileType](#) (double)
Method to set the tile type (by numeric input).
- void [setTileResource](#) ([TileResource](#))
Method to set the tile resource (by enum value).
- void [setTileResource](#) (double)
Method to set the tile resource (by numeric input).
- void [decorateTile](#) (void)
Method to decorate tile.
- void [toggleResourceOverlay](#) (void)
Method to toggle the tile resource overlay.
- void [assess](#) (void)
Method to assess the tile's resource.
- void [processEvent](#) (void)
Method to process [HexTile](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [HexTile](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- [~HexTile](#) (void)
Destructor for the [HexTile](#) class.

Public Attributes

- [TileType](#) tile_type
- [TileResource](#) tile_resource
- bool show_node
A boolean which indicates whether or not to show the tile node.
- bool show_resource
A boolean which indicates whether or not to show resource value.
- bool resource_assessed
A boolean which indicates whether or not the resource has been assessed.
- bool resource_assessment
A boolean which triggers a resource assessment notification.
- bool is_selected
A boolean which indicates whether or not the tile is selected.
- bool draw_explosion
A boolean which indicates whether or not to draw a tile explosion.
- bool decoration_cleared
A boolean which indicates if the tile decoration has been cleared.
- bool has_improvement
A boolean which indicates if tile has improvement or not.
- [TileImprovement](#) * tile_improvement_ptr
A pointer to the improvement for this tile.
- bool build_menu_open

- A boolean which indicates if the tile build menu is open.*
- `size_t explosion_frame`

The current frame of the explosion animation.
- `int frame`

The current frame of this object.
- `int credits`

The current balance of credits.
- `double position_x`

The x position of the tile.
- `double position_y`

The y position of the tile.
- `double major_radius`

The radius of the smallest bounding circle.
- `double minor_radius`

The radius of the largest inscribed circle.
- `std::string game_phase`

The current phase of the game.
- `sf::CircleShape node_sprite`

A circle shape to mark the tile node.
- `sf::ConvexShape tile_sprite`

A convex shape which represents the tile.
- `sf::ConvexShape select_outline_sprite`

A convex shape which outlines the tile when selected.
- `sf::CircleShape resource_chip_sprite`

A circle shape which represents a resource chip.
- `sf::Text resource_text`

A text representation of the resource.
- `sf::Sprite tile_decoration_sprite`

A tile decoration sprite.
- `sf::Sprite magnifying_glass_sprite`

A magnifying glass sprite.
- `std::vector< sf::Sprite > explosion_sprite_reel`

A reel of sprites for a tile explosion animation.
- `sf::RectangleShape build_menu_backing`

A backing for the tile build menu.
- `sf::Text build_menu_backing_text`

A text label for the build menu.
- `std::vector< std::vector< sf::Sprite > > build_menu_options_vec`

A vector of sprites for illustrating the tile build options.
- `std::vector< sf::Text > build_menu_options_text_vec`

A vector of text for the tile build options.

Private Member Functions

- `void __setUpNodeSprite (void)`

Helper method to set up node sprite.
- `void __setUpTileSprite (void)`

Helper method to set up tile sprite.
- `void __setUpSelectOutlineSprite (void)`

Helper method to set up select outline sprite.

- void [__setUpResourceChipSprite](#) (void)
Helper method to set up resource chip sprite.
- void [__setUpResourceText](#) (void)
Helper method to set up resource text.
- void [__setUpMagnifyingGlassSprite](#) (void)
Helper method to set up and position magnifying glass sprite.
- void [__setUpTileExplosionReel](#) (void)
Helper method to set up tile explosion sprite reel.
- void [__setUpBuildOption](#) (std::string, std::string)
Helper method to set up and position the sprite and text for a build option.
- void [__setUpDieselGeneratorBuildOption](#) (void)
Helper method to set up and position the diesel generator build option.
- void [__setUpWindTurbineBuildOption](#) (bool=false, bool=false)
Helper method to set up and position the wind turbine build option.
- void [__setUpSolarPVBuildOption](#) (bool=false)
Helper method to set up and position the solar PV array build option.
- void [__setUpTidalTurbineBuildOption](#) (void)
Helper method to set up and position the tidal turbine build option.
- void [__setUpWaveEnergyConverterBuildOption](#) (void)
Helper method to set up and position the wave energy converter build option.
- void [__setUpEnergyStorageSystemBuildOption](#) (void)
Helper method to set up and position the wave energy converter build option.
- void [__setUpBuildMenu](#) (void)
Helper method to set up and place build menu assets (drawable).
- void [__setIsSelected](#) (bool)
Helper method to set the is selected attribute (of tile and improvement).
- void [__clearDecoration](#) (void)
Helper method to clear tile decoration.
- bool [__isClicked](#) (void)
Helper method to determine if tile was clicked on.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__openBuildMenu](#) (void)
Helper method to open the tile improvement build menu.
- void [__closeBuildMenu](#) (void)
Helper method to close the tile improvement build menu.
- void [__buildSettlement](#) (void)
Helper method to build a settlement on this tile.
- void [__buildDieselGenerator](#) (void)
Helper method to build a diesel generator on this tile.
- void [__buildSolarPV](#) (void)
Helper method to build a solar PV array on this tile.
- void [__buildWindTurbine](#) (void)
Helper method to build a wind turbine on this tile.
- void [__buildTidalTurbine](#) (void)
Helper method to build a tidal turbine on this tile.
- void [__buildWaveEnergyConverter](#) (void)
Helper method to build a wave energy converter on this tile.
- void [__buildEnergyStorage](#) (void)

- Helper method to build an energy storage system on this tile.*

 - void [__sendTileSelectedMessage](#) (void)

Helper method to format and send message on tile selection.
- std::string [__getTileCoordsSubstring](#) (void)

Helper method to assemble and return tile coordinates substring.
- std::string [__getTileTypeSubstring](#) (void)

Helper method to assemble and return tile type substring.
- std::string [__getTileResourceSubstring](#) (void)

Helper method to assemble and return tile resource substring.
- std::string [__getTileImprovementSubstring](#) (void)

Helper method to assemble and return the tile improvement substring.
- std::string [__getTileOptionsSubstring](#) (void)

Helper method to assemble and return tile options substring.
- void [__sendTileStateMessage](#) (void)

Helper method to format and send tile state message.
- void [__sendAssessNeighboursMessage](#) (void)

Helper method to format and send assess neighbours message.
- void [__sendGameStateRequest](#) (void)

Helper method to format and send a game state request (message).
- void [__sendUpdateGamePhaseMessage](#) (std::string)

Helper method to format and send update game phase message.
- void [__sendCreditsSpentMessage](#) (int)

Helper method to format and send a credits spent message.
- void [__sendInsufficientCreditsMessage](#) (void)

Helper method to format and send an insufficient credits message.

Private Attributes

- sf::Event * [event_ptr](#)
A pointer to the event class.
- sf::RenderWindow * [render_window_ptr](#)
A pointer to the render window.
- [AssetsManager](#) * [assets_manager_ptr](#)
A pointer to the assets manager.
- [MessageHub](#) * [message_hub_ptr](#)
A pointer to the message hub.

4.7.1 Detailed Description

A class which defines a hex tile of the hex map.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 HexTile()

```
HexTile::HexTile (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [HexTile](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
2135 {
2136     // 1. set attributes
2137
2138     // 1.1. private
2139     this->event_ptr = event_ptr;
2140     this->render_window_ptr = render_window_ptr;
2141
2142     this->assets_manager_ptr = assets_manager_ptr;
2143     this->message_hub_ptr = message_hub_ptr;
2144
2145     // 1.2. public
2146     this->show_node = false;
2147     this->show_resource = false;
2148     this->resource_assessed = false;
2149     this->resource_assessment = false;
2150     this->is_selected = false;
2151     this->draw_explosion = false;
2152
2153     this->decoration_cleared = false;
2154     this->has_improvement = false;
2155     this->tile_improvement_ptr = NULL;
2156
2157     this->build_menu_open = false;
2158
2159     this->explosion_frame = 0;
2160
2161     this->frame = 0;
2162     this->credits = 0;
2163
2164     this->position_x = position_x;
2165     this->position_y = position_y;
2166
2167     this->major_radius = 32;
2168     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2169
2170     this->game_phase = "build settlement";
2171
2172     // 2. set up and position drawable attributes
2173     this->__setUpNodeSprite();
2174     this->__setUpTileSprite();
2175     this->__setUpSelectOutlineSprite();
2176     this->__setUpResourceChipSprite();
2177     this->__setUpResourceText();
2178     this->__setUpMagnifyingGlassSprite();
2179     this->__setUpTileExplosionReel();
2180
2181     // 3. set tile type and resource (default to none type and average)
2182     this->setTileType(TileType :: NONE_TYPE);
2183     this->setTileResource(TileResource :: AVERAGE);
```

```

2184
2185     std::cout << "HexTile constructed at " << this << std::endl;
2186
2187     return;
2188 } /* HexTile() */

```

4.7.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

2719 {
2720     if (this->tile_improvement_ptr != NULL) {
2721         delete this->tile_improvement_ptr;
2722     }
2723
2724     std::cout << "HexTile at " << this << " destroyed" << std::endl;
2725
2726     return;
2727 } /* ~HexTile() */

```

4.7.3 Member Function Documentation

4.7.3.1 __buildDieselGenerator()

```

void HexTile::__buildDieselGenerator (
    void ) [private]

```

Helper method to build a diesel generator on this tile.

```

1310 {
1311     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1312
1313     if (this->credits < build_cost) {
1314         std::cout << "Cannot build diesel generator: insufficient credits (need "
1315             << build_cost << " K)" << std::endl;
1316
1317         this->__sendInsufficientCreditsMessage();
1318         return;
1319     }
1320
1321     this->tile_improvement_ptr = new DieselGenerator(
1322         this->position_x,
1323         this->position_y,
1324         this->event_ptr,
1325         this->render_window_ptr,
1326         this->assets_manager_ptr,
1327         this->message_hub_ptr
1328     );
1329
1330     this->has_improvement = true;
1331     this->__closeBuildMenu();
1332
1333     this->__sendCreditsSpentMessage(build_cost);
1334     this->__sendTileStateMessage();
1335     this->__sendGameStateRequest();
1336
1337     return;
1338 } /* __buildDieselGenerator() */

```

4.7.3.2 __buildEnergyStorage()

```
void HexTile::__buildEnergyStorage (
    void ) [private]
```

Helper method to build an energy storage system on this tile.

```
1553 {
1554     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1555
1556     if (this->credits < build_cost) {
1557         std::cout << "Cannot build energy storage system: insufficient credits (need "
1558             << build_cost << " K)" << std::endl;
1559
1560         this->__sendInsufficientCreditsMessage();
1561         return;
1562     }
1563
1564     this->tile_improvement_ptr = new EnergyStorageSystem(
1565         this->position_x,
1566         this->position_y,
1567         this->event_ptr,
1568         this->render_window_ptr,
1569         this->assets_manager_ptr,
1570         this->message_hub_ptr
1571     );
1572
1573     this->has_improvement = true;
1574     this->__closeBuildMenu();
1575
1576     this->__sendCreditsSpentMessage(build_cost);
1577     this->__sendTileStateMessage();
1578     this->__sendGameStateRequest();
1579
1580     return;
1581 } /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

```
void HexTile::__buildSettlement (
    void ) [private]
```

Helper method to build a settlement on this tile.

```
1264 {
1265     if (this->credits < BUILD_SETTLEMENT_COST) {
1266         std::cout << "Cannot build settlement: insufficient credits (need "
1267             << BUILD_SETTLEMENT_COST << " K)" << std::endl;
1268
1269         this->__sendInsufficientCreditsMessage();
1270         return;
1271     }
1272
1273     this->__clearDecoration();
1274
1275     this->tile_improvement_ptr = new Settlement(
1276         this->position_x,
1277         this->position_y,
1278         this->event_ptr,
1279         this->render_window_ptr,
1280         this->assets_manager_ptr,
1281         this->message_hub_ptr
1282     );
1283
1284     this->has_improvement = true;
1285
1286     this->assess();
1287     this->__sendAssessNeighboursMessage();
1288
1289     this->__sendUpdateGamePhaseMessage("system management");
1290     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1291     this->__sendTileStateMessage();
1292     this->__sendGameStateRequest();
1293
1294     return;
1295 } /* __buildSettlement() */
```

4.7.3.4 __buildSolarPV()

```
void HexTile::__buildSolarPV (
    void ) [private]
```

Helper method to build a solar PV array on this tile.

```
1353 {
1354     int build_cost = SOLAR_PV_BUILD_COST;
1355
1356     if (this->tile_type == TileType :: LAKE) {
1357         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1358     }
1359
1360     if (this->credits < build_cost) {
1361         std::cout << "Cannot build solar PV array: insufficient credits (need "
1362             << build_cost << " K)" << std::endl;
1363
1364         this->__sendInsufficientCreditsMessage();
1365         return;
1366     }
1367
1368     this->tile_improvement_ptr = new SolarPV(
1369         this->position_x,
1370         this->position_y,
1371         this->event_ptr,
1372         this->render_window_ptr,
1373         this->assets_manager_ptr,
1374         this->message_hub_ptr
1375     );
1376
1377     this->has_improvement = true;
1378     this->__closeBuildMenu();
1379
1380     if (this->tile_type == TileType :: LAKE) {
1381         this->decoration_cleared = true;
1382         this->assets_manager_ptr->getSound("splash")->play();
1383     }
1384
1385     this->__sendCreditsSpentMessage(build_cost);
1386     this->__sendTileStateMessage();
1387     this->__sendGameStateRequest();
1388
1389     return;
1390 } /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

```
void HexTile::__buildTidalTurbine (
    void ) [private]
```

Helper method to build a tidal turbine on this tile.

```
1463 {
1464     int build_cost = TIDAL_TURBINE_BUILD_COST;
1465
1466     if (this->credits < build_cost) {
1467         std::cout << "Cannot build tidal turbine: insufficient credits (need "
1468             << build_cost << " K)" << std::endl;
1469
1470         this->__sendInsufficientCreditsMessage();
1471         return;
1472     }
1473
1474     this->tile_improvement_ptr = new TidalTurbine(
1475         this->position_x,
1476         this->position_y,
1477         this->event_ptr,
1478         this->render_window_ptr,
1479         this->assets_manager_ptr,
1480         this->message_hub_ptr
1481     );
1482
1483     this->has_improvement = true;
1484     this->decoration_cleared = true;
1485     this->assets_manager_ptr->getSound("splash")->play();
1486     this->__closeBuildMenu();
```

```

1487
1488     this->__sendCreditsSpentMessage(build_cost);
1489     this->__sendTileStateMessage();
1490     this->__sendGameStateRequest();
1491
1492     return;
1493 } /* __buildTidalTurbine() */

```

4.7.3.6 __buildWaveEnergyConverter()

```

void HexTile::__buildWaveEnergyConverter (
    void ) [private]

```

Helper method to build a wave energy converter on this tile.

```

1508 {
1509     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1510
1511     if (this->credits < build_cost) {
1512         std::cout << "Cannot build wave energy converter: insufficient credits (need "
1513             << build_cost << " K)" << std::endl;
1514
1515         this->__sendInsufficientCreditsMessage();
1516         return;
1517     }
1518
1519     this->tile_improvement_ptr = new WaveEnergyConverter(
1520         this->position_x,
1521         this->position_y,
1522         this->event_ptr,
1523         this->render_window_ptr,
1524         this->assets_manager_ptr,
1525         this->message_hub_ptr
1526     );
1527
1528     this->has_improvement = true;
1529     this->decoration_cleared = true;
1530     this->assets_manager_ptr->getSound("splash")->play();
1531     this->__closeBuildMenu();
1532
1533     this->__sendCreditsSpentMessage(build_cost);
1534     this->__sendTileStateMessage();
1535     this->__sendGameStateRequest();
1536
1537     return;
1538 } /* __buildWaveEnergyConverter() */

```

4.7.3.7 __buildWindTurbine()

```

void HexTile::__buildWindTurbine (
    void ) [private]

```

Helper method to build a wind turbine on this tile.

```

1405 {
1406     int build_cost = WIND_TURBINE_BUILD_COST;
1407
1408     if (
1409         (this->tile_type == TileType :: LAKE) or
1410         (this->tile_type == TileType :: OCEAN)
1411     ) {
1412         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1413     }
1414
1415     if (this->credits < build_cost) {
1416         std::cout << "Cannot build wind turbine: insufficient credits (need "
1417             << build_cost << " K)" << std::endl;
1418
1419         this->__sendInsufficientCreditsMessage();
1420         return;
1421     }

```

```

1422
1423     this->tile_improvement_ptr = new WindTurbine(
1424         this->position_x,
1425         this->position_y,
1426         this->event_ptr,
1427         this->render_window_ptr,
1428         this->assets_manager_ptr,
1429         this->message_hub_ptr
1430     );
1431
1432     this->has_improvement = true;
1433     this->__closeBuildMenu();
1434
1435     if (
1436         (this->tile_type == TileType :: LAKE) or
1437         (this->tile_type == TileType :: OCEAN)
1438     ) {
1439         this->decoration_cleared = true;
1440         this->assets_manager_ptr->getSound("splash")->play();
1441     }
1442
1443     this->__sendCreditsSpentMessage(build_cost);
1444     this->__sendTileStateMessage();
1445     this->__sendGameStateRequest();
1446
1447     return;
1448 } /* __buildWindTurbine() */

```

4.7.3.8 __clearDecoration()

```

void HexTile::__clearDecoration (
    void ) [private]

```

Helper method to clear tile decoration.

```

756 {
757     this->decoration_cleared = true;
758     this->draw_explosion = true;
759
760     switch (this->tile_type) {
761         case (TileType :: FOREST): {
762             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
763
764             break;
765         }
766
767         case (TileType :: MOUNTAINS): {
768             this->assets_manager_ptr->getSound("clear mountains tile")->play();
769
770             break;
771         }
772
773         case (TileType :: PLAINS): {
774             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
775
776             break;
777         }
778
779         default: {
780             // do nothing!
781
782             break;
783         }
784     }
785
786     return;
787 }
788
789 /* __clearDecoration() */
790 }

```


4.7.3.9 __closeBuildMenu()

```
void HexTile::__closeBuildMenu (
    void ) [private]
```

Helper method to close the tile improvement build menu.

```
1239 {
1240     if (not this->build_menu_open) {
1241         return;
1242     }
1243
1244     this->build_menu_open = false;
1245     this->assets_manager_ptr->getSound("build menu close")->play();
1246
1247     return;
1248 } /* __closeBuildMenu() */
```

4.7.3.10 __getTileCoordsSubstring()

```
std::string HexTile::__getTileCoordsSubstring (
    void ) [private]
```

Helper method to assemble and return tile coordinates substring.

Returns

Tile coordinates substring.

```
1622 {
1623     std::string coords_substring = "TILE COORDS: ";
1624     coords_substring += std::to_string(int(this->position_x - 400));
1625     coords_substring += ", ";
1626     coords_substring += std::to_string(int(this->position_y - 400));
1627     coords_substring += "\n";
1628
1629     return coords_substring;
1630 } /* __getTileCoordsSubstring() */
```

4.7.3.11 __getTileImprovementSubstring()

```
std::string HexTile::__getTileImprovementSubstring (
    void ) [private]
```

Helper method to assemble and return the tile improvement substring.

Returns

Tile improvement substring.

```
1781 {
1782     std::string improvement_substring = "TILE IMPROVEMENT: ";
1783
1784     if (this->has_improvement) {
1785         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1786         improvement_substring += "\n";
1787     }
1788
1789     else {
1790         improvement_substring += "NONE\n";
1791     }
1792
1793     return improvement_substring;
1794 } /* __getTileImprovementSubstring() */
```

4.7.3.12 __getTileOptionsSubstring()

```
std::string HexTile::__getTileOptionsSubstring (
    void ) [private]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

```
1811 {
1812     //          32 char x 17 line console "-----\n";
1813     std::string options_substring          = "      **** TILE OPTIONS **** \n";
1814     options_substring                     += " \n";
1815
1816     if (this->game_phase == "build settlement") {
1817         if (
1818             (this->tile_type != TileType :: OCEAN) and
1819             (this->tile_type != TileType :: LAKE)
1820         ) {
1821             options_substring += "[B]:  BUILD SETTLEMENT (";
1822             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
1823             options_substring += " K)";
1824         }
1825     }
1826
1827
1828     else if (this->game_phase == "system management") {
1829         if (this->has_improvement) {
1830             /*
1831             options_substring.clear();
1832             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
1833             */
1834         }
1835
1836
1837         else if (not this->resource_assessed) {
1838             options_substring += "[A]:  ASSESS RESOURCE (";
1839             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
1840             options_substring += " K)\n";
1841         }
1842
1843
1844         else if (
1845             (not this->decoration_cleared) and
1846             (this->tile_type != TileType :: OCEAN) and
1847             (this->tile_type != TileType :: LAKE)
1848         ) {
1849             options_substring += "[C]:  CLEAR TILE (";
1850
1851             switch (this->tile_type) {
1852                 case (TileType :: FOREST): {
1853                     options_substring += std::to_string(CLEAR_FOREST_COST);
1854                     break;
1855                 }
1856
1857
1858                 case (TileType :: MOUNTAINS): {
1859                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
1860                     break;
1861                 }
1862
1863                 case (TileType :: PLAINS): {
1864                     options_substring += std::to_string(CLEAR_PLAINS_COST);
1865                     break;
1866                 }
1867
1868                 default: {
1869                     //do nothing!
1870                     break;
1871                 }
1872             }
1873
1874             options_substring += " K)\n";
1875         }
1876     }
1877 }
```

```

1882
1883
1884         else if (
1885             (this->decoration_cleared) or
1886             (this->tile_type == TileType :: OCEAN) or
1887             (this->tile_type == TileType :: LAKE)
1888         ) {
1889             options_substring += "[B]: OPEN BUILD MENU\n";
1890         }
1891     }
1892
1893
1894     else if (this->game_phase == "victory") {
1895         options_substring += "      **** VICTORY ****      \n";
1896     }
1897
1898
1899     else {
1900         options_substring += "      **** LOSS ****      \n";
1901     }
1902
1903     return options_substring;
1904 } /* __getTileOptionsString() */

```

4.7.3.13 __getTileResourceSubstring()

```

std::string HexTile::__getTileResourceSubstring (
    void ) [private]

```

Helper method to assemble and return tile resource substring.

Returns

Tile resource substring.

```

1711 {
1712     std::string resource_substring = "TILE RESOURCE: ";
1713
1714     if (this->resource_assessed) {
1715         switch (this->tile_resource) {
1716             case (TileResource :: POOR): {
1717                 resource_substring += "POOR\n";
1718
1719                 break;
1720             }
1721
1722
1723             case (TileResource :: BELOW_AVERAGE): {
1724                 resource_substring += "BELOW AVERAGE\n";
1725
1726                 break;
1727             }
1728
1729
1730             case (TileResource :: AVERAGE): {
1731                 resource_substring += "AVERAGE\n";
1732
1733                 break;
1734             }
1735
1736
1737             case (TileResource :: ABOVE_AVERAGE): {
1738                 resource_substring += "ABOVE AVERAGE\n";
1739
1740                 break;
1741             }
1742
1743
1744             case (TileResource :: GOOD): {
1745                 resource_substring += "GOOD\n";
1746
1747                 break;
1748             }
1749
1750
1751             default: {

```

```

1752             resource_substring += "???\n";
1753
1754             break;
1755         }
1756     }
1757 }
1758
1759 else {
1760     resource_substring += "???\n";
1761 }
1762
1763 return resource_substring;
1764 } /* __getTileResourceSubstring() */

```

4.7.3.14 __getTileTypeSubstring()

```

std::string HexTile::__getTileTypeSubstring (
    void ) [private]

```

Helper method to assemble and return tile type substring.

Returns

Tile type substring.

```

1647 {
1648     std::string type_substring = "TILE TYPE: ";
1649
1650     switch (this->tile_type) {
1651         case (TileType :: FOREST): {
1652             type_substring += "FOREST\n";
1653
1654             break;
1655         }
1656
1657         case (TileType :: LAKE): {
1658             type_substring += "LAKE\n";
1659
1660             break;
1661         }
1662
1663         case (TileType :: MOUNTAINS): {
1664             type_substring += "MOUNTAINS\n";
1665
1666             break;
1667         }
1668
1669         case (TileType :: OCEAN): {
1670             type_substring += "OCEAN\n";
1671
1672             break;
1673         }
1674
1675         case (TileType :: PLAINS): {
1676             type_substring += "PLAINS\n";
1677
1678             break;
1679         }
1680
1681         default: {
1682             type_substring += "???\n";
1683
1684             break;
1685         }
1686     }
1687
1688     return type_substring;
1689 } /* __getTileTypeSubstring() */

```

4.7.3.15 __handleKeyPressEvents()

```
void HexTile::__handleKeyPressEvents (
    void ) [private]
```

Helper method to handle key press events.

```
839 {
840     if (not this->is_selected) {
841         return;
842     }
843
844
845     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
846         this->__setIsSelected(false);
847     }
848
849
850     if (this->build_menu_open) {
851         switch (this->tile_type) {
852             case (TileType :: FOREST): {
853                 switch (this->event_ptr->key.code) {
854                     case (sf::Keyboard::D): {
855                         this->__buildDieselGenerator();
856
857                         break;
858                     }
859
860                     case (sf::Keyboard::S): {
861                         this->__buildSolarPV();
862
863                         break;
864                     }
865
866                     case (sf::Keyboard::W): {
867                         this->__buildWindTurbine();
868
869                         break;
870                     }
871
872                     case (sf::Keyboard::E): {
873                         this->__buildEnergyStorage();
874
875                         break;
876                     }
877
878                     default: {
879                         // do nothing!
880
881                         break;
882                     }
883                 }
884             }
885
886             break;
887         }
888
889         case (TileType :: LAKE): {
890             switch (this->event_ptr->key.code) {
891                 case (sf::Keyboard::S): {
892                     this->__buildSolarPV();
893
894                     break;
895                 }
896
897                 case (sf::Keyboard::W): {
898                     this->__buildWindTurbine();
899
900                     break;
901                 }
902
903                 default: {
904                     // do nothing!
905
906                     break;
907                 }
908             }
909
910             break;
911         }
912
913         break;
914     }
915
916     break;
```

```

917     }
918
919
920     case (TileType :: MOUNTAINS): {
921         switch (this->event_ptr->key.code) {
922             case (sf::Keyboard::D): {
923                 this->__buildDieselGenerator();
924
925                 break;
926             }
927
928
929             case (sf::Keyboard::S): {
930                 this->__buildSolarPV();
931
932                 break;
933             }
934
935
936             case (sf::Keyboard::W): {
937                 this->__buildWindTurbine();
938
939                 break;
940             }
941
942
943             case (sf::Keyboard::E): {
944                 this->__buildEnergyStorage();
945
946                 break;
947             }
948
949
950             default: {
951                 // do nothing!
952
953                 break;
954             }
955         }
956
957         break;
958     }
959
960
961     case (TileType :: OCEAN): {
962         switch (this->event_ptr->key.code) {
963             case (sf::Keyboard::W): {
964                 this->__buildWindTurbine();
965
966                 break;
967             }
968
969
970             case (sf::Keyboard::T): {
971                 this->__buildTidalTurbine();
972
973                 break;
974             }
975
976
977             case (sf::Keyboard::A): {
978                 this->__buildWaveEnergyConverter();
979
980                 break;
981             }
982
983
984             default: {
985                 // do nothing!
986
987                 break;
988             }
989         }
990
991         break;
992     }
993
994
995     case (TileType :: PLAINS): {
996         switch (this->event_ptr->key.code) {
997             case (sf::Keyboard::D): {
998                 this->__buildDieselGenerator();
999
1000                 break;
1001             }
1002
1003

```

```

1004         case (sf::Keyboard::S): {
1005             this->__buildSolarPV();
1006
1007             break;
1008         }
1009
1010
1011         case (sf::Keyboard::W): {
1012             this->__buildWindTurbine();
1013
1014             break;
1015         }
1016
1017
1018         case (sf::Keyboard::E): {
1019             this->__buildEnergyStorage();
1020
1021             break;
1022         }
1023
1024
1025         default: {
1026             // do nothing!
1027
1028             break;
1029         }
1030     }
1031
1032     break;
1033 }
1034
1035
1036     default: {
1037         //do nothing!
1038
1039         break;
1040     }
1041 }
1042
1043
1044
1045 if (this->game_phase == "build settlement") {
1046     if (
1047         (this->tile_type != TileType :: OCEAN) and
1048         (this->tile_type != TileType :: LAKE)
1049     ) {
1050         if (this->event_ptr->key.code == sf::Keyboard::B) {
1051             this->__buildSettlement();
1052         }
1053     }
1054 }
1055
1056
1057 else if (this->game_phase == "system management") {
1058     if (this->has_improvement) {
1059         // will be caught by this->tile_improvement_ptr->processEvent();
1060     }
1061
1062
1063     else if (not this->resource_assessed) {
1064         if (this->event_ptr->key.code == sf::Keyboard::A) {
1065             if (this->credits < RESOURCE_ASSESSMENT_COST) {
1066                 std::cout << "Cannot assess resource: insufficient credits (need "
1067                     << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
1068
1069                 this->__sendInsufficientCreditsMessage();
1070             }
1071
1072             else {
1073                 this->assess();
1074                 this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1075                 this->__sendTileStateMessage();
1076                 this->__sendGameStateRequest();
1077             }
1078         }
1079     }
1080
1081
1082     else if (
1083         (not this->decoration_cleared) and
1084         (this->tile_type != TileType :: OCEAN) and
1085         (this->tile_type != TileType :: LAKE)
1086     ) {
1087         if (this->event_ptr->key.code == sf::Keyboard::C) {
1088             int clear_cost = 0;
1089
1090             switch (this->tile_type) {

```

```

1091         case (TileType :: FOREST): {
1092             clear_cost = CLEAR_FOREST_COST;
1093
1094             break;
1095         }
1096
1097         case (TileType :: MOUNTAINS): {
1098             clear_cost = CLEAR_MOUNTAINS_COST;
1099
1100             break;
1101         }
1102
1103         case (TileType :: PLAINS): {
1104             clear_cost = CLEAR_PLAINS_COST;
1105
1106             break;
1107         }
1108
1109         default: {
1110             // do nothing!
1111
1112             break;
1113         }
1114     }
1115
1116     if (this->credits < clear_cost) {
1117         std::cout << "Cannot clear tile: insufficient credits (need "
1118             << clear_cost << " K)" << std::endl;
1119
1120         this->__sendInsufficientCreditsMessage();
1121     }
1122
1123     else {
1124         this->__clearDecoration();
1125         this->__sendCreditsSpentMessage(clear_cost);
1126         this->__sendTileStateMessage();
1127         this->__sendGameStateRequest();
1128     }
1129 }
1130
1131 else if (
1132     (this->decoration_cleared) or
1133     (this->tile_type == TileType :: OCEAN) or
1134     (this->tile_type == TileType :: LAKE)
1135 ) {
1136     if (this->event_ptr->key.code == sf::Keyboard::B) {
1137         this->__openBuildMenu();
1138     }
1139 }
1140
1141 return;
1142 }
1143
1144 /* __handleKeyPressEvents() */

```

4.7.3.16 __handleMouseButtonEvents()

```

void HexTile::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

1163 {
1164     switch (this->event_ptr->mouseButton.button) {
1165         case (sf::Mouse::Left): {
1166             if (this->__isClicked()) {
1167                 std::cout << "Tile (" << this->position_x << ", " <<
1168                     this->position_y << ") was selected" << std::endl;
1169
1170                 this->__setIsSelected(true);
1171
1172                 this->__sendTileSelectedMessage();
1173                 this->__sendTileStateMessage();
1174                 this->__sendGameStateRequest();
1175             }

```



```

1176
1177         else {
1178             this->__setIsSelected(false);
1179         }
1180
1181         break;
1182     }
1183
1184
1185     case (sf::Mouse::Right): {
1186         this->__setIsSelected(false);
1187
1188         break;
1189     }
1190
1191
1192     default: {
1193         // do nothing!
1194
1195         break;
1196     }
1197 }
1198
1199 return;
1200 } /* __handleMouseButtonEvents() */

```

4.7.3.17 __isClicked()

```

bool HexTile::__isClicked (
    void ) [private]

```

Helper method to determine if tile was clicked on.

Returns

Boolean indicating whether or not tile was clicked on.

```

807 {
808     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
809
810     double mouse_x = mouse_position.x;
811     double mouse_y = mouse_position.y;
812
813     double distance = sqrt(
814         pow(this->position_x - mouse_x, 2) +
815         pow(this->position_y - mouse_y, 2)
816     );
817
818     if (distance < this->minor_radius) {
819         return true;
820     }
821     else {
822         return false;
823     }
824 } /* __isClicked() */

```

4.7.3.18 __openBuildMenu()

```

void HexTile::__openBuildMenu (
    void ) [private]

```

Helper method to open the tile improvement build menu.

```

1215 {
1216     if (this->build_menu_open) {
1217         return;
1218     }
1219
1220     this->build_menu_open = true;
1221     this->assets_manager_ptr->getSound("build menu open")->play();
1222
1223     return;
1224 } /* __openBuildMenu() */

```

4.7.3.19 __sendAssessNeighboursMessage()

```
void HexTile::__sendAssessNeighboursMessage (
    void ) [private]
```

Helper method to format and send assess neighbours message.

```
1966 {
1967     Message assess_neighbours_message;
1968
1969     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
1970     assess_neighbours_message.subject = "assess neighbours";
1971
1972     this->message_hub_ptr->sendMessage(assess_neighbours_message);
1973
1974     std::cout << "Assess neighbours message sent by " << this << std::endl;
1975
1976     return;
1977 } /* __sendAssessNeighboursMessage() */
```

4.7.3.20 __sendCreditsSpentMessage()

```
void HexTile::__sendCreditsSpentMessage (
    int credits_spent ) [private]
```

Helper method to format and send a credits spent message.

Parameters

<i>credits_spent</i>	The number of credits that were spent.
----------------------	--

```
2049 {
2050     Message credits_spent_message;
2051
2052     credits_spent_message.channel = GAME_CHANNEL;
2053     credits_spent_message.subject = "credits spent";
2054
2055     credits_spent_message.int_payload["credits spent"] = credits_spent;
2056
2057     this->message_hub_ptr->sendMessage(credits_spent_message);
2058
2059     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2060         << std::endl;
2061     return;
2062 } /* __sendCreditsSpentMessage() */
```

4.7.3.21 __sendGameStateRequest()

```
void HexTile::__sendGameStateRequest (
    void ) [private]
```

Helper method to format and send a game state request (message).

```
1992 {
1993     Message game_state_request;
1994
1995     game_state_request.channel = GAME_CHANNEL;
1996     game_state_request.subject = "state request";
1997
1998     this->message_hub_ptr->sendMessage(game_state_request);
1999
2000     std::cout << "Game state request message sent by " << this << std::endl;
2001     return;
2002 } /* __sendGameStateRequest() */
```

4.7.3.22 __sendInsufficientCreditsMessage()

```
void HexTile::__sendInsufficientCreditsMessage (
    void ) [private]
```

Helper method to format and send an insufficient credits message.

```
2077 {
2078     Message insufficient_credits_message;
2079
2080     insufficient_credits_message.channel = GAME_CHANNEL;
2081     insufficient_credits_message.subject = "insufficient credits";
2082
2083     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2084
2085     std::cout << "Insufficient credits message sent by " << this << std::endl;
2086
2087     return;
2088 } /* __sendInsufficientCreditsMessage() */
```

4.7.3.23 __sendTileSelectedMessage()

```
void HexTile::__sendTileSelectedMessage (
    void ) [private]
```

Helper method to format and send message on tile selection.

```
1596 {
1597     Message tile_selected_message;
1598
1599     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1600     tile_selected_message.subject = "tile selected";
1601
1602     this->message_hub_ptr->sendMessage(tile_selected_message);
1603
1604     return;
1605 } /* __sendTileSelectedMessage() */
```

4.7.3.24 __sendTileStateMessage()

```
void HexTile::__sendTileStateMessage (
    void ) [private]
```

Helper method to format and send tile state message.

```
1919 {
1920     Message tile_state_message;
1921
1922     tile_state_message.channel = TILE_STATE_CHANNEL;
1923     tile_state_message.subject = "tile state";
1924
1925
1926     //          32 char x 17 line console "-----\n";
1927     std::string console_string = "      **** TILE INFO ****\n";
1928     console_string += "      \n";
1929
1930     console_string += this->__getTileCoordsSubstring();
1931     console_string += "      \n";
1932
1933     console_string += this->__getTileTypeSubstring();
1934
1935     if (not this->has_improvement) {
1936         console_string += this->__getTileResourceSubstring();
1937     }
1938
1939     console_string += this->__getTileImprovementSubstring();
1940     console_string += "      \n";
1941
1942     console_string += this->__getTileOptionsSubstring();
```

```

1943
1944
1945     tile_state_message.string_payload["console string"] = console_string;
1946
1947     this->message_hub_ptr->sendMessage(tile_state_message);
1948
1949     std::cout << "Tile state message sent by " << this << std::endl;
1950     return;
1951 }    /* __sendTileStateMessage() */

```

4.7.3.25 __sendUpdateGamePhaseMessage()

```

void HexTile::__sendUpdateGamePhaseMessage (
    std::string game_phase ) [private]

```

Helper method to format and send update game phase message.

Parameters

<i>game_phase</i>	The updated game phase.
-------------------	-------------------------

```

2019 {
2020     Message update_game_phase_message;
2021
2022     update_game_phase_message.channel = GAME_CHANNEL;
2023     update_game_phase_message.subject = "update game phase";
2024
2025     update_game_phase_message.string_payload["game phase"] = game_phase;
2026
2027     this->message_hub_ptr->sendMessage(update_game_phase_message);
2028
2029     std::cout << "Update game phase message sent by " << this << std::endl;
2030
2031     return;
2032 }    /* __sendUpdateGamePhaseMessage() */

```

4.7.3.26 __setIsSelected()

```

void HexTile::__setIsSelected (
    bool is_selected ) [private]

```

Helper method to set the is selected attribute (of tile and improvement).

Parameters

<i>is_selected</i>	The value to set the is selected attribute to.
--------------------	--

```

729 {
730     this->is_selected = is_selected;
731
732     if (this->tile_improvement_ptr != NULL) {
733         this->tile_improvement_ptr->is_selected = is_selected;
734     }
735
736     if ((not is_selected) and this->build_menu_open) {
737         this->__closeBuildMenu();
738     }
739
740     return;
741 }    /* __toggleIsSelected() */

```

4.7.3.27 __setResourceText()

```
void HexTile::__setResourceText (
    void ) [private]
```

Helper method to set up resource text.

```
159 {
160     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
161
162     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
163
164     if (this->resource_assessed) {
165         switch (this->tile_resource) {
166             case (TileResource :: POOR): {
167                 this->resource_text.setString("-2");
168                 this->resource_text.setFillColor(MONOCROME_TEXT_RED);
169
170                 break;
171             }
172
173             case (TileResource :: BELOW_AVERAGE): {
174                 this->resource_text.setString("-1");
175                 this->resource_text.setFillColor(MONOCROME_TEXT_RED);
176
177                 break;
178             }
179
180             case (TileResource :: AVERAGE): {
181                 this->resource_text.setString("+0");
182
183                 break;
184             }
185
186             case (TileResource :: ABOVE_AVERAGE): {
187                 this->resource_text.setString("+1");
188                 this->resource_text.setFillColor(MONOCROME_TEXT_GREEN);
189
190                 break;
191             }
192
193             case (TileResource :: GOOD): {
194                 this->resource_text.setString("+2");
195                 this->resource_text.setFillColor(MONOCROME_TEXT_GREEN);
196
197                 break;
198             }
199
200             default: {
201                 this->resource_text.setString("");
202
203                 break;
204             }
205         }
206     }
207
208     else {
209         this->resource_text.setString("");
210     }
211
212     this->resource_text.setCharacterSize(20);
213
214     this->resource_text.setOrigin(
215         this->resource_text.getLocalBounds().width / 2,
216         this->resource_text.getLocalBounds().height / 2
217     );
218
219     this->resource_text.setPosition(
220         this->position_x,
221         this->position_y - 4
222     );
223
224     this->resource_text.setOutlineThickness(1);
225     this->resource_text.setOutlineColor(sf::Color(0, 0, 0, 255));
226
227     return;
228 } /* __setResourceText() */
```

4.7.3.28 __setUpBuildMenu()

```
void HexTile::__setUpBuildMenu (
    void ) [private]
```

Helper method to set up and place build menu assets (drawable).

```
632 {
633     this->build_menu_options_vec.clear();
634     this->build_menu_options_text_vec.clear();
635
636     // 1. set up and place build menu backing and text
637     this->build_menu_backing.setSize(sf::Vector2f(600, 256));
638     this->build_menu_backing.setOrigin(300, 128);
639     this->build_menu_backing.setPosition(400, 400);
640     this->build_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
641     this->build_menu_backing.setOutlineColor(MENU_FRAME_GREY);
642     this->build_menu_backing.setOutlineThickness(4);
643
644     this->build_menu_backing_text.setString("**** BUILD MENU ****");
645     this->build_menu_backing_text.setFont(
646         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
647     );
648     this->build_menu_backing_text.setCharacterSize(16);
649     this->build_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
650     this->build_menu_backing_text.setOrigin(
651         this->build_menu_backing_text.getLocalBounds().width / 2, 0
652     );
653     this->build_menu_backing_text.setPosition(400, 400 - 128 + 4);
654
655     // 2. set up and place build menu option sprites and text
656     switch (this->tile_type) {
657         case (TileType :: FOREST): {
658             this->__setUpDieselGeneratorBuildOption();
659             this->__setUpSolarPVBuildOption();
660             this->__setUpWindTurbineBuildOption();
661             this->__setUpEnergyStorageSystemBuildOption();
662
663             break;
664         }
665
666         case (TileType :: LAKE): {
667             this->__setUpSolarPVBuildOption(true);
668             this->__setUpWindTurbineBuildOption(true);
669
670             break;
671         }
672
673         case (TileType :: MOUNTAINS): {
674             this->__setUpDieselGeneratorBuildOption();
675             this->__setUpSolarPVBuildOption();
676             this->__setUpWindTurbineBuildOption();
677             this->__setUpEnergyStorageSystemBuildOption();
678
679             break;
680         }
681
682         case (TileType :: OCEAN): {
683             this->__setUpWindTurbineBuildOption(false, true);
684             this->__setUpTidalTurbineBuildOption();
685             this->__setUpWaveEnergyConverterBuildOption();
686
687             break;
688         }
689
690         case (TileType :: PLAINS): {
691             this->__setUpDieselGeneratorBuildOption();
692             this->__setUpSolarPVBuildOption();
693             this->__setUpWindTurbineBuildOption();
694             this->__setUpEnergyStorageSystemBuildOption();
695
696             break;
697         }
698
699         default: {
700             // do nothing!
701
702             break;
703         }
704     }
705 }
```

```

710
711     return;
712 } /* __setUpBuildMenu() */

```

4.7.3.29 __setUpBuildOption()

```

void HexTile::__setUpBuildOption (
    std::string texture_key,
    std::string option_string ) [private]

```

Helper method to set up and position the sprite and text for a build option.

Parameters

<i>texture_key</i>	The key for the appropriate illustration asset for the build option.
<i>option_string</i>	A string for the build option.

```

323 {
324     size_t n_options = this->build_menu_options_vec.size();
325
326     // 1. set up option sprite(s)
327     this->build_menu_options_vec.push_back({});
328
329     if (not texture_key.empty()) {
330         sf::Sprite texture_sheet(
331             *(this->assets_manager_ptr->getTexture(texture_key))
332         );
333
334         int sheet_height = texture_sheet.getLocalBounds().height;
335         int n_subrects = sheet_height / 64;
336
337         for (int i = 0; i < n_subrects; i++) {
338             this->build_menu_options_vec.back().push_back(
339                 sf::Sprite(
340                     *(this->assets_manager_ptr->getTexture(texture_key)),
341                     sf::IntRect(0, i * 64, 64, 64)
342                 )
343             );
344
345             this->build_menu_options_vec.back().back().setOrigin(
346                 this->build_menu_options_vec.back().back().getLocalBounds().width / 2,
347                 this->build_menu_options_vec.back().back().getLocalBounds().height
348             );
349
350             this->build_menu_options_vec.back().back().setPosition(
351                 400 - 300 + 75 + n_options * 150,
352                 400 - 32
353             );
354         }
355     }
356
357     else {
358         this->build_menu_options_vec.back().push_back(sf::Sprite());
359     }
360
361
362     // 2. set up option text
363     this->build_menu_options_text_vec.push_back(
364         sf::Text(
365             option_string,
366             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
367             16
368         )
369     );
370
371     this->build_menu_options_text_vec.back().setOrigin(
372         this->build_menu_options_text_vec.back().getLocalBounds().width / 2,
373         0
374     );
375
376     this->build_menu_options_text_vec.back().setPosition(
377         400 - 300 + 75 + n_options * 150,
378         400 - 16 - 4

```

```

379     );
380
381     this->build_menu_options_text_vec.back().setFillColor(MONochrome_TEXT_GREEN);
382
383     return;
384 } /* __setUpBuildOption() */

```

4.7.3.30 __setUpDieselGeneratorBuildOption()

```

void HexTile::__setUpDieselGeneratorBuildOption (
    void ) [private]

```

Helper method to set up and position the diesel generator build option.

```

399 {
400     // 1. set up option sprite(s)
401     std::string texture_key = "diesel generator";
402
403     // 2. set up option string (up to 16 chars wide)
404     // -----
405     std::string diesel_generator_string = "DIESEL GENERATOR\n";
406     diesel_generator_string += " \n";
407     diesel_generator_string += "CAPACITY: 100 kW\n";
408     diesel_generator_string += "COST: ";
409     diesel_generator_string += std::to_string(DIESEL_GENERATOR_BUILD_COST);
410     diesel_generator_string += " K\n\n";
411     diesel_generator_string += "BUILD: [D] \n";
412
413     // 3. call general method
414     this->__setUpBuildOption(texture_key, diesel_generator_string);
415
416     return;
417 } /* __setUpDieselGeneratorBuildOption() */

```

4.7.3.31 __setUpEnergyStorageSystemBuildOption()

```

void HexTile::__setUpEnergyStorageSystemBuildOption (
    void ) [private]

```

Helper method to set up and position the wave energy converter build option.

```

599 {
600     // 1. set up option sprite(s)
601     std::string texture_key = "energy storage system";
602
603     // 2. set up option string (up to 16 chars wide)
604     // -----
605     std::string energy_storage_system_string = "ENERGY STORAGE \n";
606     energy_storage_system_string += " \n";
607     energy_storage_system_string += "CAPCTY: 500 kWh\n";
608     energy_storage_system_string += "COST: ";
609     energy_storage_system_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
610     energy_storage_system_string += " K\n\n";
611     energy_storage_system_string += "BUILD: [E] \n";
612
613     // 3. call general method
614     this->__setUpBuildOption(texture_key, energy_storage_system_string);
615
616     return;
617 } /* __setUpEnergyStorageSystemBuildOption() */

```


4.7.3.32 __setUpMagnifyingGlassSprite()

```
void HexTile::__setUpMagnifyingGlassSprite (
    void ) [private]
```

Helper method to set up and position magnifying glass sprite.

```
243 {
244     this->magnifying_glass_sprite.setTexture(
245         *(this->assets_manager_ptr->getTexture("magnifying_glass_64x64_1"))
246     );
247
248     this->magnifying_glass_sprite.setOrigin(
249         this->magnifying_glass_sprite.getLocalBounds().width / 2,
250         this->magnifying_glass_sprite.getLocalBounds().height / 2
251     );
252
253     this->magnifying_glass_sprite.setPosition(
254         this->position_x,
255         this->position_y
256     );
257
258     return;
259 } /* __setUpMagnifyingGlassSprite() */
```

4.7.3.33 __setUpNodeSprite()

```
void HexTile::__setUpNodeSprite (
    void ) [private]
```

Helper method to set up node sprite.

```
34 {
35     this->node_sprite.setRadius(4);
36
37     this->node_sprite.setOrigin(
38         this->node_sprite.getLocalBounds().width / 2,
39         this->node_sprite.getLocalBounds().height / 2
40     );
41
42     this->node_sprite.setPosition(this->position_x, this->position_y);
43
44     this->node_sprite.setFillColor(sf::Color(255, 0, 0, 255));
45
46     return;
47 } /* __setUpNodeSprite() */
```

4.7.3.34 __setUpResourceChipSprite()

```
void HexTile::__setUpResourceChipSprite (
    void ) [private]
```

Helper method to set up resource chip sprite.

```
132 {
133     this->resource_chip_sprite.setRadius(2 * this->minor_radius / 3);
134
135     this->resource_chip_sprite.setOrigin(
136         this->resource_chip_sprite.getLocalBounds().width / 2,
137         this->resource_chip_sprite.getLocalBounds().height / 2
138     );
139
140     this->resource_chip_sprite.setPosition(this->position_x, this->position_y);
141
142     this->resource_chip_sprite.setFillColor(RESOURCE_CHIP_GREY);
143
144     return;
145 } /* __setUpResourceChip() */
```

4.7.3.35 __setUpSelectOutlineSprite()

```
void HexTile::__setUpSelectOutlineSprite (
    void ) [private]
```

Helper method to set up select outline sprite.

```
96 {
97     int n_points = 6;
98
99     this->select_outline_sprite.setPointCount(n_points);
100
101     for (int i = 0; i < n_points; i++) {
102         this->select_outline_sprite.setPoint(
103             i,
104             sf::Vector2f(
105                 this->position_x + this->major_radius * cos((30 + 60 * i) * (M_PI / 180)),
106                 this->position_y + this->major_radius * sin((30 + 60 * i) * (M_PI / 180))
107             )
108         );
109     }
110
111     this->select_outline_sprite.setOutlineThickness(4);
112     this->select_outline_sprite.setOutlineColor(MONOCHROME_TEXT_RED);
113
114     this->select_outline_sprite.setFill(sf::Color(0, 0, 0, 0));
115
116     return;
117 } /* __setUpSelectOutline() */
```

4.7.3.36 __setUpSolarPVBuildOption()

```
void HexTile::__setUpSolarPVBuildOption (
    bool is_lake = false ) [private]
```

Helper method to set up and position the solar PV array build option.

Parameters

<i>is_lake</i>	If being built on a lake.
----------------	---------------------------

```
487 {
488     // 1. set up option sprite(s)
489     std::string texture_key = "solar PV array";
490
491     // 2. set up option string (up to 16 chars wide)
492     int build_cost = SOLAR_PV_BUILD_COST;
493     if (is_lake) {
494         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
495     }
496
497     //
498     std::string solar_PV_string = "-----\n"
499     solar_PV_string             = " SOLAR PV ARRAY \n";
500     solar_PV_string             += " \n";
501     solar_PV_string             += "CAPACITY: 100 kW\n";
502     solar_PV_string             += "COST:      ";
503     solar_PV_string             += std::to_string(build_cost);
504     solar_PV_string             += " K";
505
506     if (is_lake) {
507         solar_PV_string += "\n** LAKE BUILD **\n\n";
508     }
509     else {
510         solar_PV_string += "\n\n\n";
511     }
512
513     solar_PV_string             += "BUILD:      [S]  \n";
514
515     // 3. call general method
516     this->__setUpBuildOption(texture_key, solar_PV_string);
517
518     return;
519 }
```

```
518 }    /* __setUpSolarPVBuildOption() */
```

4.7.3.37 __setUpTidalTurbineBuildOption()

```
void HexTile::__setUpTidalTurbineBuildOption (
    void ) [private]
```

Helper method to set up and position the tidal turbine build option.

```
533 {
534     // 1. set up option sprite(s)
535     std::string texture_key = "tidal turbine";
536
537     // 2. set up option string (up to 16 chars wide)
538     // -----\n"
539     std::string tidal_turbine_string = " TIDAL TURBINE \n";
540     tidal_turbine_string += " \n";
541     tidal_turbine_string += "CAPACITY: 100 kW\n";
542     tidal_turbine_string += "COST: ";
543     tidal_turbine_string += std::to_string(TIDAL_TURBINE_BUILD_COST);
544     tidal_turbine_string += " K\n\n";
545     tidal_turbine_string += "BUILD: [T] \n";
546
547     // 3. call general method
548     this->__setUpBuildOption(texture_key, tidal_turbine_string);
549
550     return;
551 }    /* __setUpTidalTurbineBuildOption() */
```

4.7.3.38 __setUpTileExplosionReel()

```
void HexTile::__setUpTileExplosionReel (
    void ) [private]
```

Helper method to set up tile explosion sprite reel.

```
274 {
275     for (int i = 0; i < 4; i++) {
276         for (int j = 0; j < 4; j++) {
277             this->explosion_sprite_reel.push_back(
278                 sf::Sprite(
279                     *(this->assets_manager_ptr->getTexture("tile clear explosion")),
280                     sf::IntRect(j * 64, i * 64, 64, 64)
281                 )
282             );
283
284             this->explosion_sprite_reel.back().setOrigin(
285                 this->explosion_sprite_reel.back().getLocalBounds().width / 2,
286                 this->explosion_sprite_reel.back().getLocalBounds().height / 2
287             );
288
289             this->explosion_sprite_reel.back().setPosition(
290                 this->position_x,
291                 this->position_y
292             );
293         }
294     }
295
296     return;
297 }    /* __setUpTileExplosionReel() */
```

4.7.3.39 __setUpTileSprite()

```
void HexTile::__setUpTileSprite (
    void ) [private]
```

Helper method to set up tile sprite.

```
62 {
63     int n_points = 6;
64
65     this->tile_sprite.setPointCount(n_points);
66
67     for (int i = 0; i < n_points; i++) {
68         this->tile_sprite.setPoint(
69             i,
70             sf::Vector2f(
71                 this->position_x + this->major_radius * cos((30 + 60 * i) * (M_PI / 180)),
72                 this->position_y + this->major_radius * sin((30 + 60 * i) * (M_PI / 180))
73             )
74         );
75     }
76
77     this->tile_sprite.setOutlineThickness(1);
78     this->tile_sprite.setOutlineColor(sf::Color(175, 175, 175, 255));
79
80     return;
81 } /* __setUpTileSprite() */
```

4.7.3.40 __setUpWaveEnergyConverterBuildOption()

```
void HexTile::__setUpWaveEnergyConverterBuildOption (
    void ) [private]
```

Helper method to set up and position the wave energy converter build option.

```
566 {
567     // 1. set up option sprite(s)
568     std::string texture_key = "wave energy converter";
569
570     // 2. set up option string (up to 16 chars wide)
571     // -----\n"
572     std::string wave_energy_converter_string = "WAVE ENERGY CVTR\n";
573     wave_energy_converter_string += "          \n";
574     wave_energy_converter_string += "CAPACITY: 100 kW\n";
575     wave_energy_converter_string += "COST:      ";
576     wave_energy_converter_string += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
577     wave_energy_converter_string += " K\n\n";
578     wave_energy_converter_string += "BUILD:    [A]  \n";
579
580     // 3. call general method
581     this->__setUpBuildOption(texture_key, wave_energy_converter_string);
582
583     return;
584 } /* __setUpWaveEnergyConverterBuildOption() */
```

4.7.3.41 __setUpWindTurbineBuildOption()

```
void HexTile::__setUpWindTurbineBuildOption (
    bool is_lake = false,
    bool is_ocean = false ) [private]
```

Helper method to set up and position the wind turbine build option.

Parameters

<i>is_lake</i>	If being built on a lake tile.
<i>is_ocean</i>	If being built on an ocean tile.

```

436 {
437     // 1. set up option sprite(s)
438     std::string texture_key = "wind turbine";
439
440     // 2. set up option string (up to 16 chars wide)
441     int build_cost = WIND_TURBINE_BUILD_COST;
442     if (is_lake or is_ocean) {
443         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
444     }
445
446     //
447     std::string wind_turbine_string      = "  WIND TURBINE  \n";
448     wind_turbine_string                  += "  \n";
449     wind_turbine_string                  += "CAPACITY: 100 kW\n";
450     wind_turbine_string                  += "COST:      ";
451     wind_turbine_string                  += std::to_string(build_cost);
452     wind_turbine_string                  += " K";
453
454     if (is_lake) {
455         wind_turbine_string += "\n** LAKE BUILD **\n\n";
456     }
457     else if (is_ocean) {
458         wind_turbine_string += "\n* OCEAN BUILD * \n\n";
459     }
460     else {
461         wind_turbine_string += "\n\n\n";
462     }
463
464     wind_turbine_string                  += "BUILD:      [W]  \n";
465
466     // 3. call general method
467     this->__setUpBuildOption(texture_key, wind_turbine_string);
468
469     return;
470 } /* __setUpWindTurbineBuildOption() */

```

4.7.3.42 assess()

```

void HexTile::assess (
    void )

```

Method to assess the tile's resource.

```

2509 {
2510     this->resource_assessed = true;
2511     this->resource_assessment = true;
2512
2513     this->assets_manager_ptr->getSound("resource assessment")->play();
2514
2515     this->__setResourceText();
2516     this->__sendTileStateMessage();
2517
2518     return;
2519 } /* assess() */

```

4.7.3.43 decorateTile()

```

void HexTile::decorateTile (
    void )

```

Method to decorate tile.

```

2387 {

```

```

2388     switch (this->tile_type) {
2389     case (TileType :: FOREST): {
2390         this->tile_decoration_sprite.setTexture(
2391             *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2392         );
2393
2394         break;
2395     }
2396
2397     case (TileType :: LAKE): {
2398         this->tile_decoration_sprite.setTexture(
2399             *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2400         );
2401
2402         break;
2403     }
2404
2405     case (TileType :: MOUNTAINS): {
2406         this->tile_decoration_sprite.setTexture(
2407             *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2408         );
2409
2410         break;
2411     }
2412
2413     case (TileType :: OCEAN): {
2414         this->tile_decoration_sprite.setTexture(
2415             *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2416         );
2417
2418         break;
2419     }
2420
2421     case (TileType :: PLAINS): {
2422         this->tile_decoration_sprite.setTexture(
2423             *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2424         );
2425
2426         break;
2427     }
2428
2429     default: {
2430         // do nothing!
2431
2432         break;
2433     }
2434 }
2435
2436
2437 if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
2438     this->tile_decoration_sprite.setOrigin(
2439         this->tile_decoration_sprite.getLocalBounds().width / 2,
2440         this->tile_decoration_sprite.getLocalBounds().height / 2
2441     );
2442
2443     this->tile_decoration_sprite.setPosition(
2444         this->position_x,
2445         this->position_y
2446     );
2447
2448     if ((double)rand() / RAND_MAX > 0.5) {
2449         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2450     }
2451 }
2452
2453 else {
2454     this->tile_decoration_sprite.setOrigin(
2455         this->tile_decoration_sprite.getLocalBounds().width / 2,
2456         this->tile_decoration_sprite.getLocalBounds().height
2457     );
2458
2459     this->tile_decoration_sprite.setPosition(
2460         this->position_x,
2461         this->position_y + 12
2462     );
2463
2464     if ((double)rand() / RAND_MAX > 0.5) {
2465         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2466     }
2467 }
2468
2469 return;
2470 } /* decorateTile(void) */

```

4.7.3.44 draw()

```
void HexTile::draw (
    void )
```

Method to draw the hex tile to the render window. To be called once per frame.

```
2614 {
2615     // 1. draw hex
2616     this->render_window_ptr->draw(this->tile_sprite);
2617
2618     // 2. draw node
2619     if (this->show_node) {
2620         this->render_window_ptr->draw(this->node_sprite);
2621     }
2622
2623     // 3. draw tile decoration
2624     if (not this->decoration_cleared) {
2625         this->render_window_ptr->draw(this->tile_decoration_sprite);
2626     }
2627
2628     // 4. draw tile improvement
2629     if (this->has_improvement) {
2630         if (not this->tile_improvement_ptr->just_built) {
2631             this->tile_improvement_ptr->draw();
2632         }
2633     }
2634
2635     // 5. draw resource
2636     if (this->show_resource) {
2637         this->render_window_ptr->draw(this->resource_chip_sprite);
2638         this->render_window_ptr->draw(this->resource_text);
2639     }
2640
2641     // 6. draw selection outline
2642     if (this->is_selected) {
2643         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2644
2645         outline_colour.a =
2646             255 * pow(cos((M_PI * this->frame) / (1.5 * FRAMES_PER_SECOND)), 2);
2647
2648         this->select_outline_sprite.setOutlineColor(outline_colour);
2649
2650         this->render_window_ptr->draw(this->select_outline_sprite);
2651     }
2652
2653     // 7. draw resource assessment notification
2654     if (this->resource_assessment) {
2655         int alpha = this->magnifying_glass_sprite.getColor().a;
2656
2657         alpha -= 0.05 * FRAMES_PER_SECOND;
2658         if (alpha < 0) {
2659             alpha = 0;
2660             this->resource_assessment = false;
2661         }
2662
2663         this->magnifying_glass_sprite.setColor(
2664             sf::Color(255, 255, 255, alpha)
2665         );
2666
2667         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2668     }
2669
2670     // 8. draw explosion, then settlement placement
2671     if (this->draw_explosion) {
2672         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2673
2674         if (this->frame % (FRAMES_PER_SECOND / 10) == 0) {
2675             this->explosion_frame++;
2676         }
2677
2678         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2679             this->draw_explosion = false;
2680         }
2681     }
2682
2683     else if (this->has_improvement) {
2684         if (this->tile_improvement_ptr->just_built) {
2685             this->tile_improvement_ptr->draw();
2686         }
2687     }
2688
2689     // 9. build menu
2690     if (this->build_menu_open) {
2691         this->render_window_ptr->draw(this->build_menu_backing);
```

```

2692         this->render_window_ptr->draw(this->build_menu_backing_text);
2693
2694         for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2695             for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2696                 this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2697             }
2698             this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2699         }
2700     }
2701
2702     this->frame++;
2703     return;
2704 } /* draw() */

```

4.7.3.45 processEvent()

```

void HexTile::processEvent (
    void )

```

Method to process [HexTile](#). To be called once per event.

```

2534 {
2535     // 1. process TileImprovement events
2536     if (this->tile_improvement_ptr != NULL) {
2537         this->tile_improvement_ptr->processEvent();
2538     }
2539
2540     // 2. process HexTile events
2541     if (this->event_ptr->type == sf::Event::KeyPressed) {
2542         this->__handleKeyPressEvents();
2543     }
2544
2545     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2546         this->__handleMouseButtonEvents();
2547     }
2548
2549     return;
2550 } /* processEvent() */

```

4.7.3.46 processMessage()

```

void HexTile::processMessage (
    void )

```

Method to process [HexTile](#). To be called once per message.

```

2565 {
2566     // 1. process TileImprovement messages
2567     if (this->tile_improvement_ptr != NULL) {
2568         this->tile_improvement_ptr->processMessage();
2569     }
2570
2571     // 2. process HexTile messages
2572     if (this->is_selected) {
2573         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2574             Message game_state_message = this->message_hub_ptr->receiveMessage(
2575                 GAME_STATE_CHANNEL
2576             );
2577
2578             if (game_state_message.subject == "game state") {
2579                 this->credits = game_state_message.int_payload["credits"];
2580                 this->game_phase = game_state_message.string_payload["game phase"];
2581
2582                 if (this->tile_improvement_ptr != NULL) {
2583                     this->tile_improvement_ptr->credits = this->credits;
2584                     this->tile_improvement_ptr->game_phase = this->game_phase;
2585                 }
2586
2587                 std::cout << "Game state message received by " << this << std::endl;
2588                 this->__sendTileStateMessage();
2589                 this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);

```



```

2590         }
2591     }
2592
2593     std::cout << "Current credits (HexTile): " << this->credits << " K" <<
2594         std::endl;
2595 }
2596
2597 return;
2598 } /* processMessage() */

```

4.7.3.47 setTileResource() [1/2]

```

void HexTile::setTileResource (
    double input_value )

```

Method to set the tile resource (by numeric input).

Parameters

<i>input_value</i>	A numerical input in the closed interval [0, 1].
--------------------	--

```

2336 {
2337     // 1. check input
2338     if (input_value < 0 or input_value > 1) {
2339         std::string error_str = "ERROR HexTile::setTileResource() given input value is ";
2340         error_str += "not in the closed interval [0, 1]";
2341
2342         #ifdef _WIN32
2343             std::cout << error_str << std::endl;
2344         #endif /* _WIN32 */
2345
2346         throw std::runtime_error(error_str);
2347     }
2348
2349     // 2. convert input value to tile resource
2350     TileResource tile_resource;
2351
2352     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2353         tile_resource = TileResource :: POOR;
2354     }
2355     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2356         tile_resource = TileResource :: BELOW_AVERAGE;
2357     }
2358     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2359         tile_resource = TileResource :: AVERAGE;
2360     }
2361     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2362         tile_resource = TileResource :: ABOVE_AVERAGE;
2363     }
2364     else {
2365         tile_resource = TileResource :: GOOD;
2366     }
2367
2368     // 3. call alternate method
2369     this->setTileResource(tile_resource);
2370
2371     return;
2372 } /* setTileResource(double) */

```

4.7.3.48 setTileResource() [2/2]

```

void HexTile::setTileResource (
    TileResource tile_resource )

```

Method to set the tile resource (by enum value).

Parameters

<i>tile_resource</i>	The resource (TileResource) value to attribute to the tile.
----------------------	---

```

2314 {
2315     this->tile_resource = tile_resource;
2316     this->__setResourceText();
2317
2318     return;
2319 } /* setTileResource(TileResource) */

```

4.7.3.49 setTileType() [1/2]

```

void HexTile::setTileType (
    double input_value )

```

Method to set the tile type (by numeric input).

Parameters

<i>input_value</i>	A numerical input in the closed interval [0, 1].
--------------------	--

```

2264 {
2265     // 1. check input
2266     if (input_value < 0 or input_value > 1) {
2267         std::string error_str = "ERROR HexTile::setTileType() given input value is ";
2268         error_str += "not in the closed interval [0, 1]";
2269
2270         #ifdef _WIN32
2271             std::cout << error_str << std::endl;
2272         #endif /* _WIN32 */
2273
2274         throw std::runtime_error(error_str);
2275     }
2276
2277     // 2. convert input value to tile type
2278     TileType tile_type;
2279
2280     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2281         tile_type = TileType :: LAKE;
2282     }
2283     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2284         tile_type = TileType :: PLAINS;
2285     }
2286     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2287         tile_type = TileType :: FOREST;
2288     }
2289     else {
2290         tile_type = TileType :: MOUNTAINS;
2291     }
2292
2293     // 3. call alternate method
2294     this->setTileType(tile_type);
2295
2296     return;
2297 } /* setTileType(double) */

```

4.7.3.50 setTileType() [2/2]

```

void HexTile::setTileType (
    TileType tile_type )

```

Method to set the tile type (by enum value).

Parameters

<i>tile_type</i>	The type (TileType) to set the tile to.
------------------	---

```

2203 {
2204     this->tile_type = tile_type;
2205
2206     switch (this->tile_type) {
2207         case (TileType :: FOREST): {
2208             this->tile_sprite.setFillColor(FOREST_GREEN);
2209
2210             break;
2211         }
2212
2213         case (TileType :: LAKE): {
2214             this->tile_sprite.setFillColor(LAKE_BLUE);
2215
2216             break;
2217         }
2218
2219         case (TileType :: MOUNTAINS): {
2220             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2221
2222             break;
2223         }
2224
2225         case (TileType :: OCEAN): {
2226             this->tile_sprite.setFillColor(OCEAN_BLUE);
2227
2228             break;
2229         }
2230
2231         case (TileType :: PLAINS): {
2232             this->tile_sprite.setFillColor(PLAINS_YELLOW);
2233
2234             break;
2235         }
2236
2237         default: {
2238             // do nothing!
2239
2240             break;
2241         }
2242     }
2243
2244     this->__setUpBuildMenu();
2245
2246     return;
2247 } /* setTileType(TileType) */

```

4.7.3.51 toggleResourceOverlay()

```

void HexTile::toggleResourceOverlay(
    void )

```

Method to toggle the tile resource overlay.

```

2485 {
2486     if (this->show_resource) {
2487         this->show_resource = false;
2488     }
2489     else {
2490         this->show_resource = true;
2491     }
2492
2493     return;
2494 } /* toggleResourceOverlay() */

```

4.7.4 Member Data Documentation

4.7.4.1 assets_manager_ptr

```
AssetsManager* HexTile::assets_manager_ptr [private]
```

A pointer to the assets manager.

4.7.4.2 build_menu_backing

```
sf::RectangleShape HexTile::build_menu_backing
```

A backing for the tile build menu.

4.7.4.3 build_menu_backing_text

```
sf::Text HexTile::build_menu_backing_text
```

A text label for the build menu.

4.7.4.4 build_menu_open

```
bool HexTile::build_menu_open
```

A boolean which indicates if the tile build menu is open.

4.7.4.5 build_menu_options_text_vec

```
std::vector<sf::Text> HexTile::build_menu_options_text_vec
```

A vector of text for the tile build options.

4.7.4.6 build_menu_options_vec

```
std::vector<std::vector<sf::Sprite> > HexTile::build_menu_options_vec
```

A vector of sprites for illustrating the tile build options.

4.7.4.7 credits

```
int HexTile::credits
```

The current balance of credits.

4.7.4.8 decoration_cleared

```
bool HexTile::decoration_cleared
```

A boolean which indicates if the tile decoration has been cleared.

4.7.4.9 draw_explosion

```
bool HexTile::draw_explosion
```

A boolean which indicates whether or not to draw a tile explosion.

4.7.4.10 event_ptr

```
sf::Event* HexTile::event_ptr [private]
```

A pointer to the event class.

4.7.4.11 explosion_frame

```
size_t HexTile::explosion_frame
```

The current frame of the explosion animation.

4.7.4.12 explosion_sprite_reel

```
std::vector<sf::Sprite> HexTile::explosion_sprite_reel
```

A reel of sprites for a tile explosion animation.

4.7.4.13 frame

```
int HexTile::frame
```

The current frame of this object.

4.7.4.14 game_phase

```
std::string HexTile::game_phase
```

The current phase of the game.

4.7.4.15 has_improvement

```
bool HexTile::has_improvement
```

A boolean which indicates if tile has improvement or not.

4.7.4.16 is_selected

```
bool HexTile::is_selected
```

A boolean which indicates whether or not the tile is selected.

4.7.4.17 magnifying_glass_sprite

```
sf::Sprite HexTile::magnifying_glass_sprite
```

A magnifying glass sprite.

4.7.4.18 major_radius

```
double HexTile::major_radius
```

The radius of the smallest bounding circle.

4.7.4.19 message_hub_ptr

```
MessageHub* HexTile::message_hub_ptr [private]
```

A pointer to the message hub.

4.7.4.20 minor_radius

```
double HexTile::minor_radius
```

The radius of the largest inscribed circle.

4.7.4.21 node_sprite

```
sf::CircleShape HexTile::node_sprite
```

A circle shape to mark the tile node.

4.7.4.22 position_x

```
double HexTile::position_x
```

The x position of the tile.

4.7.4.23 position_y

```
double HexTile::position_y
```

The y position of the tile.

4.7.4.24 render_window_ptr

```
sf::RenderWindow* HexTile::render_window_ptr [private]
```

A pointer to the render window.

4.7.4.25 resource_assessed

```
bool HexTile::resource_assessed
```

A boolean which indicates whether or not the resource has been assessed.

4.7.4.26 resource_assessment

```
bool HexTile::resource_assessment
```

A boolean which triggers a resource assessment notification.

4.7.4.27 resource_chip_sprite

```
sf::CircleShape HexTile::resource_chip_sprite
```

A circle shape which represents a resource chip.

4.7.4.28 resource_text

```
sf::Text HexTile::resource_text
```

A text representation of the resource.

4.7.4.29 select_outline_sprite

```
sf::ConvexShape HexTile::select_outline_sprite
```

A convex shape which outlines the tile when selected.

4.7.4.30 show_node

```
bool HexTile::show_node
```

A boolean which indicates whether or not to show the tile node.

4.7.4.31 show_resource

```
bool HexTile::show_resource
```

A boolean which indicates whether or not to show resource value.

4.7.4.32 tile_decoration_sprite

```
sf::Sprite HexTile::tile_decoration_sprite
```

A tile decoration sprite.

4.7.4.33 tile_improvement_ptr

```
TileImprovement* HexTile::tile_improvement_ptr
```

A pointer to the improvement for this tile.

4.7.4.34 tile_resource

```
TileResource HexTile::tile_resource
```

4.7.4.35 tile_sprite

```
sf::ConvexShape HexTile::tile_sprite
```

A convex shape which represents the tile.

4.7.4.36 tile_type

```
TileType HexTile::tile_type
```

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

- header/[HexTile.h](#)
- source/[HexTile.cpp](#)

4.8 Message Struct Reference

A structure which defines a standard message format.

```
#include <MessageHub.h>
```

Public Attributes

- `std::string channel = ""`
A string identifying the appropriate channel for this message.
- `std::string subject = ""`
A string describing the message subject.
- `std::map< std::string, bool > bool_payload = {}`
A boolean payload.
- `std::map< std::string, int > int_payload = {}`
A vector payload.
- `std::map< std::string, double > double_payload = {}`
A vector payload.
- `std::map< std::string, std::string > string_payload = {}`
A string payload.

4.8.1 Detailed Description

A structure which defines a standard message format.

4.8.2 Member Data Documentation

4.8.2.1 bool_payload

```
std::map<std::string, bool> Message::bool_payload = {}
```

A boolean payload.

4.8.2.2 channel

```
std::string Message::channel = ""
```

A string identifying the appropriate channel for this message.

4.8.2.3 double_payload

```
std::map<std::string, double> Message::double_payload = {}
```

A vector payload.

4.8.2.4 int_payload

```
std::map<std::string, int> Message::int_payload = {}
```

A vector payload.

4.8.2.5 string_payload

```
std::map<std::string, std::string> Message::string_payload = {}
```

A string payload.

4.8.2.6 subject

```
std::string Message::subject = ""
```

A string describing the message subject.

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

- header/ESC_core/[MessageHub.h](#)

4.9 MessageHub Class Reference

A class which acts as a central hub for inter-object message traffic.

```
#include <MessageHub.h>
```

Public Member Functions

- [MessageHub](#) (void)
Constructor for the [MessageHub](#) class.
- bool [hasTraffic](#) (void)
Method to determine if there remains any message traffic.
- void [addChannel](#) (std::string)
Method to add channel to message map.
- void [removeChannel](#) (std::string)
Method to remove channel from message map.
- void [sendMessage](#) ([Message](#))
Method to send a message to the message map. Channels are implemented in a first in, first out manner (i.e. message queue).
- bool [isEmpty](#) (std::string)
Method to check if channel is empty.
- [Message](#) [receiveMessage](#) (std::string)
Method to receive the first message in the channel. Channels are implemented in a first in, first out manner (i.e. message queue).
- void [popMessage](#) (std::string)
Method to pop first message off of the given channel. Channels are implemented in a first in, first out manner (i.e. message queue).
- void [clearMessages](#) (void)
Method to clear messages from the [MessageHub](#).
- void [clear](#) (void)
Method to clear the [MessageHub](#).
- [~MessageHub](#) (void)
Destructor for the [MessageHub](#) class.

Private Attributes

- std::map< std::string, std::list< [Message](#) > > [message_map](#)
A map <string, list of [Message](#)> for sending and receiving messages. Here the key is the channel, and each channel maintains a list (history) of messages.

4.9.1 Detailed Description

A class which acts as a central hub for inter-object message traffic.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 MessageHub()

```
MessageHub::MessageHub (
    void )
```

Constructor for the [MessageHub](#) class.

```
46 {
47     //...
48
49     std::cout << "MessageHub constructed at " << this << std::endl;
50
51     return;
52 } /* MessageHub() */
```

4.9.2.2 ~MessageHub()

```
MessageHub::~MessageHub (
    void )
```

Destructor for the [MessageHub](#) class.

```
393 {
394     this->clear();
395
396     std::cout << "MessageHub at " << this << " destroyed" << std::endl;
397
398     return;
399 } /* ~MessageHub() */
```

4.9.3 Member Function Documentation

4.9.3.1 addChannel()

```
void MessageHub::addChannel (
    std::string channel )
```

Method to add channel to message map.

Parameters

<i>channel</i>	The key for the message channel being added.
----------------	--

```
97 {
98     // 1. check if channel is in map (if so, throw error)
99     if (this->message_map.count(channel) > 0) {
100         std::string error_str = "ERROR MessageHub::addChannel() channel ";
101         error_str += channel;
102         error_str += " is already in message map";
103
104         #ifdef _WIN32
105             std::cout << error_str << std::endl;
106         #endif /* _WIN32 */
107
108         throw std::runtime_error(error_str);
109     }
110
111     // 2. add channel to map
112     this->message_map[channel] = {};
113
114     std::cout << "Channel " << channel << " added to message hub" << std::endl;
115
116     return;
117 } /* addChannel() */
```

4.9.3.2 clear()

```
void MessageHub::clear (
    void )
```

Method to clear the [MessageHub](#).

```
373 {
374
375     this->clearMessages();
```

```

376     this->message_map.clear();
377
378     return;
379 } /* clear() */

```

4.9.3.3 clearMessages()

```

void MessageHub::clearMessages (
    void )

```

Method to clear messages from the [MessageHub](#).

```

347 {
348     std::map<std::string, std::list<Message>::iterator map_iter;
349     for (
350         map_iter = this->message_map.begin();
351         map_iter != this->message_map.end();
352         map_iter++
353     ) {
354         map_iter->second.clear();
355     }
356
357     return;
358 } /* clearMessages() */

```

4.9.3.4 hasTraffic()

```

bool MessageHub::hasTraffic (
    void )

```

Method to determine if there remains any message traffic.

```

67 {
68     std::map<std::string, std::list<Message>::iterator map_iter;
69     for (
70         map_iter = this->message_map.begin();
71         map_iter != this->message_map.end();
72         map_iter++
73     ) {
74         if (not map_iter->second.empty()) {
75             return true;
76         }
77     }
78
79     return false;
80 } /* hasTraffic() */

```

4.9.3.5 isEmpty()

```

bool MessageHub::isEmpty (
    std::string channel )

```

Method to check if channel is empty.

Parameters

<i>channel</i>	The key for the message channel being checked.
----------------	--

Returns

A boolean indicating whether the channel is empty or not.

```

212 {
213     // 1. check if channel is in map (if not, throw error)
214     if (this->message_map.count(channel) <= 0) {
215         std::string error_str = "ERROR MessageHub::isEmpty() channel ";
216         error_str += channel;
217         error_str += " is not in message map";
218
219         #ifdef _WIN32
220             std::cout << error_str << std::endl;
221         #endif /* _WIN32 */
222
223         throw std::runtime_error(error_str);
224     }
225
226     if (this->message_map[channel].empty()) {
227         return true;
228     }
229     else {
230         return false;
231     }
232 } /* isEmpty() */

```

4.9.3.6 popMessage()

```

void MessageHub::popMessage (
    std::string channel )

```

Method to pop first message off of the given channel. Channels are implemented in a first in, first out manner (i.e. message queue).

Parameters

<i>channel</i>	The key for the message channel being popped.
----------------	---

```

301 {
302     // 1. check if channel is in map (if not, throw error)
303     if (this->message_map.count(channel) <= 0) {
304         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
305         error_str += channel;
306         error_str += " is not in message map";
307
308         #ifdef _WIN32
309             std::cout << error_str << std::endl;
310         #endif /* _WIN32 */
311
312         throw std::runtime_error(error_str);
313     }
314
315     // 2. check if channel is empty (if so, throw error)
316     if (this->message_map[channel].empty()) {
317         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
318         error_str += channel;
319         error_str += " is empty";
320
321         #ifdef _WIN32
322             std::cout << error_str << std::endl;
323         #endif /* _WIN32 */
324
325         throw std::runtime_error(error_str);
326     }
327
328     // 3. pop message
329     this->message_map[channel].pop_front();
330
331     return;
332 } /* popMessage() */

```

4.9.3.7 receiveMessage()

```
Message MessageHub::receiveMessage (
    std::string channel )
```

Method to receive the first message in the channel. Channels are implemented in a first in, first out manner (i.e. message queue).

Parameters

<i>channel</i>	The key for the message channel being received from.
----------------	--

Returns

The first message in the given channel.

```
252 {
253     // 1. check if channel is in map (if not, throw error)
254     if (this->message_map.count(channel) <= 0) {
255         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
256         error_str += channel;
257         error_str += " is not in message map";
258
259         #ifdef _WIN32
260             std::cout << error_str << std::endl;
261         #endif /* _WIN32 */
262
263         throw std::runtime_error(error_str);
264     }
265
266     // 2. check if channel is empty (if so, throw error)
267     if (this->message_map[channel].empty()) {
268         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
269         error_str += channel;
270         error_str += " is empty";
271
272         #ifdef _WIN32
273             std::cout << error_str << std::endl;
274         #endif /* _WIN32 */
275
276         throw std::runtime_error(error_str);
277     }
278
279     // 3. receive message
280     Message message = this->message_map[channel].front();
281
282     return message;
283 } /* receiveMessage() */
```

4.9.3.8 removeChannel()

```
void MessageHub::removeChannel (
    std::string channel )
```

Method to remove channel from message map.

Parameters

<i>channel</i>	The key for the message channel being removed.
----------------	--

```
134 {
135     // 1. check if channel is in map (if not, throw error)
136     if (this->message_map.count(channel) <= 0) {
137         std::string error_str = "ERROR MessageHub::removeChannel() channel ";
```



```

138         error_str += channel;
139         error_str += " is not in message map";
140
141         #ifdef _WIN32
142             std::cout << error_str << std::endl;
143         #endif /* _WIN32 */
144
145         throw std::runtime_error(error_str);
146     }
147
148     // 2. remove channel from map
149     this->message_map[channel].clear();
150     this->message_map.erase(channel);
151
152     std::cout << "Channel " << channel << " removed from message hub" << std::endl;
153
154     return;
155 } /* removeChannel() */

```

4.9.3.9 sendMessage()

```

void MessageHub::sendMessage (
    Message message )

```

Method to send a message to the message map. Channels are implemented in a first in, first out manner (i.e. message queue).

Parameters

<i>message</i>	The message to be sent.
----------------	-------------------------

```

173 {
174     // 1. check if channel is in map (if not, throw error)
175     std::string channel = message.channel;
176
177     if (this->message_map.count(channel) <= 0) {
178         std::string error_str = "ERROR MessageHub::sendMessage() channel ";
179         error_str += channel;
180         error_str += " is not in message map";
181
182         #ifdef _WIN32
183             std::cout << error_str << std::endl;
184         #endif /* _WIN32 */
185
186         throw std::runtime_error(error_str);
187     }
188
189     // 2. send message to message map
190     this->message_map[channel].push_back(message);
191
192     return;
193 } /* sendMessage() */

```

4.9.4 Member Data Documentation

4.9.4.1 message_map

```
std::map<std::string, std::list<Message> > MessageHub::message_map [private]
```

A map <string, list of [Message](#)> for sending and receiving messages. Here the key is the channel, and each channel maintains a list (history) of messages.

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

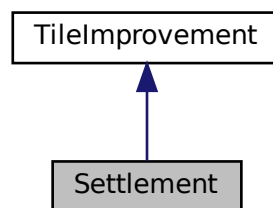
- [header/ESC_core/MessageHub.h](#)
- [source/ESC_core/MessageHub.cpp](#)

4.10 Settlement Class Reference

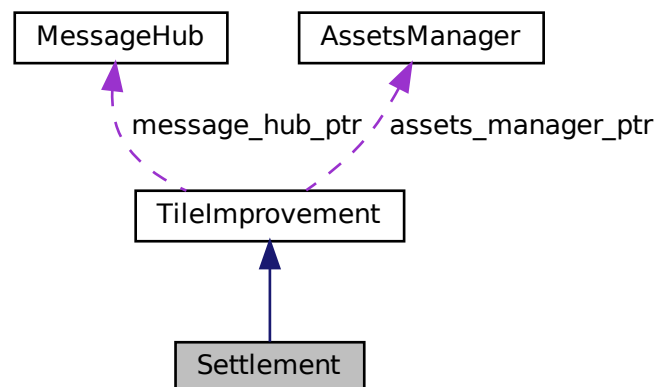
A settlement class (child class of [TileImprovement](#)).

```
#include <Settlement.h>
```

Inheritance diagram for Settlement:



Collaboration diagram for Settlement:



Public Member Functions

- [Settlement](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [Settlement](#) class.

- void [processEvent](#) (void)
Method to process [Settlement](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [Settlement](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~Settlement](#) (void)
Destructor for the [Settlement](#) class.

Public Attributes

- bool [skip_smoke_processing](#)
A boolean which indicates whether or not to skip smoke processing.
- double [smoke_da](#)
The per frame delta in smoke particle alpha value.
- double [smoke_dx](#)
The per frame delta in smoke particle x position.
- double [smoke_dy](#)
The per frame delta in smoke particle y position.
- double [smoke_prob](#)
The probability of spawning a new smoke prob in any given frame.
- std::list< sf::Sprite > [smoke_sprite_list](#)
A list of smoke sprite (for chimney animation).

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.10.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Settlement()

```
Settlement::Settlement (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [Settlement](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
167 :
168 TileImprovement (
169     position_x,
170     position_y,
171     event_ptr,
172     render_window_ptr,
173     assets_manager_ptr,
174     message_hub_ptr
175 )
176 {
177     // 1. set attributes
178
179     // 1.1. private
180     //...
181
182     // 1.2. public
183     this->tile_improvement_type = TileImprovementType :: SETTLEMENT;
184
185     this->skip_smoke_processing = true;
186
187     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
188     this->smoke_dx = 5 * SECONDS_PER_FRAME;
189     this->smoke_dy = -10 * SECONDS_PER_FRAME;
190     this->smoke_prob = 8 * SECONDS_PER_FRAME;
191
192     this->smoke_sprite_list = {};
193
194     this->tile_improvement_string = "SETTLEMENT";
195
196     this->__setUpTileImprovementSpriteStatic();
197
198     std::cout << "Settlement constructed at " << this << std::endl;
199
200     return;
201 } /* Settlement() */
```

4.10.2.2 ~Settlement()

```
Settlement::~Settlement (
    void ) [virtual]
```

Destructor for the [Settlement](#) class.

```
356 {
357     std::cout << "Settlement at " << this << " destroyed" << std::endl;
358
359     return;
360 } /* ~Settlement() */
```

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

```
void Settlement::__handleKeyPressEvents (
    void ) [private], [virtual]
```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```
69 {
70     switch (this->event_ptr->key.code) {
71         //...
72
73         default: {
74             // do nothing!
75
76             break;
77         }
78     }
79 }
80
81 return;
82 } /* __handleKeyPressEvents() */
```

4.10.3.2 __handleMouseButtonEvents()

```
void Settlement::__handleMouseButtonEvents (
    void ) [private], [virtual]
```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
97 {
98     switch (this->event_ptr->mouseButton.button) {
99         case (sf::Mouse::Left): {
100             //...
101
102             break;
103         }
104
105         case (sf::Mouse::Right): {
106             //...
107
108             break;
109         }
110
111         default: {
112             // do nothing!
113
114             break;
115         }
116     }
117 }
118
119 return;
120 } /* __handleMouseButtonEvents() */
```

4.10.3.3 __setUpTileImprovementSpriteStatic()

```
void Settlement::__setUpTileImprovementSpriteStatic (
    void ) [private]
```

Helper method to set up tile improvement sprite (static).

```
34 {
35     this->tile_improvement_sprite_static.setTexture(
36         *(this->assets_manager_ptr->getTexture("brick_house_64x64_1"))
37     );
38
39     this->tile_improvement_sprite_static.setOrigin(
40         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
41         this->tile_improvement_sprite_static.getLocalBounds().height
42     );
43
44     this->tile_improvement_sprite_static.setPosition(
45         this->position_x,
46         this->position_y - 32
47     );
48
49     this->tile_improvement_sprite_static.setColor(
50         sf::Color(255, 255, 255, 0)
51     );
52
53     return;
54 } /* __setUpTileImprovementSpriteStatic() */
```

4.10.3.4 draw()

```
void Settlement::draw (
    void ) [virtual]
```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```
261 {
262     // 1. if just built, call base method and return
263     if (this->just_built) {
264         TileImprovement :: draw();
265
266         return;
267     }
268
269     // 2. draw static sprite and chimney smoke effects
270     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
271
272     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
273
274     double alpha = 255;
275
276     while (iter != this->smoke_sprite_list.end()) {
277         this->render_window_ptr->draw(*iter);
278
279         if (not this->skip_smoke_processing) {
280             alpha = (*iter).getColor().a;
281
282             alpha -= this->smoke_da;
283
284             if (alpha <= 0) {
285                 iter = this->smoke_sprite_list.erase(iter);
286                 continue;
287             }
288
289             (*iter).setColor(sf::Color(255, 255, 255, alpha));
290
291             (*iter).move(
292                 this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
293                 this->smoke_dy
294             );
295
296             (*iter).rotate(0.5 * (((double)rand() / RAND_MAX)));
297         }
298     }
```

```

299         iter++;
300     }
301
302
303     if (not this->skip_smoke_processing) {
304         if ((double)rand() / RAND_MAX < smoke_prob) {
305             this->smoke_sprite_list.push_back(
306                 sf::Sprite(
307                     *(this->assets_manager_ptr->getTexture("steam / smoke")),
308                     sf::IntRect(0, 8, 8, 8)
309                 )
310             );
311
312             this->smoke_sprite_list.back().setOrigin(
313                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
314                 this->smoke_sprite_list.back().getLocalBounds().height / 2
315             );
316
317             this->smoke_sprite_list.back().setPosition(
318                 this->position_x + 9,
319                 this->position_y - 33
320             );
321         }
322     }
323
324
325     if (this->is_selected) {
326         if (this->skip_smoke_processing) {
327             this->skip_smoke_processing = false;
328         }
329
330         else {
331             this->skip_smoke_processing = true;
332         }
333     }
334
335     else {
336         this->skip_smoke_processing = false;
337     }
338
339     this->frame++;
340     return;
341 } /* draw() */

```

4.10.3.5 processEvent()

```

void Settlement::processEvent (
    void ) [virtual]

```

Method to process [Settlement](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```

216 {
217     if (this->event_ptr->type == sf::Event::KeyPressed) {
218         this->__handleKeyPressEvents();
219     }
220
221     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
222         this->__handleMouseButtonEvents();
223     }
224
225     return;
226 } /* processEvent() */

```

4.10.3.6 processMessage()

```

void Settlement::processMessage (
    void ) [virtual]

```

Method to process [Settlement](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
241 {  
242     //...  
243  
244     return;  
245 } /* processMessage() */
```

4.10.4 Member Data Documentation

4.10.4.1 skip_smoke_processing

```
bool Settlement::skip_smoke_processing
```

A boolean which indicates whether or not to skip smoke processing.

4.10.4.2 smoke_da

```
double Settlement::smoke_da
```

The per frame delta in smoke particle alpha value.

4.10.4.3 smoke_dx

```
double Settlement::smoke_dx
```

The per frame delta in smoke particle x position.

4.10.4.4 smoke_dy

```
double Settlement::smoke_dy
```

The per frame delta in smoke particle y position.

4.10.4.5 smoke_prob

```
double Settlement::smoke_prob
```

The probability of spawning a new smoke prob in any given frame.

4.10.4.6 smoke_sprite_list

```
std::list<sf::Sprite> Settlement::smoke_sprite_list
```

A list of smoke sprite (for chimney animation).

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

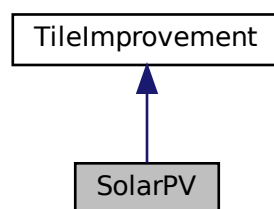
- header/[Settlement.h](#)
- source/[Settlement.cpp](#)

4.11 SolarPV Class Reference

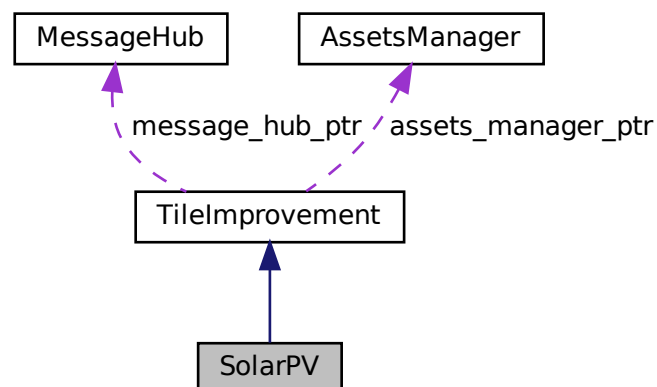
A settlement class (child class of [TileImprovement](#)).

```
#include <SolarPV.h>
```

Inheritance diagram for SolarPV:



Collaboration diagram for SolarPV:



Public Member Functions

- [SolarPV](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [SolarPV](#) class.
- void [processEvent](#) (void)
Method to process [SolarPV](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [SolarPV](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~SolarPV](#) (void)
Destructor for the [SolarPV](#) class.

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.11.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SolarPV()

```
SolarPV::SolarPV (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [SolarPV](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```

167 :
168 TileImprovement (
169     position_x,
170     position_y,
171     event_ptr,
172     render_window_ptr,
173     assets_manager_ptr,
174     message_hub_ptr
175 )
176 {
177     // 1. set attributes
178
179     // 1.1. private
180     //...
181
182     // 1.2. public
183     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
184
185     this->is_running = false;
186
187     this->tile_improvement_string = "SOLAR PV ARRAY";
188
189     this->__setUpTileImprovementSpriteStatic();
190
191     std::cout << "SolarPV constructed at " << this << std::endl;
192
193     return;
194 } /* SolarPV() */

```

4.11.2.2 ~SolarPV()

```

SolarPV::~SolarPV (
    void ) [virtual]

```

Destructor for the [SolarPV](#) class.

```

283 {
284     std::cout << "SolarPV at " << this << " destroyed" << std::endl;
285
286     return;
287 } /* ~SolarPV() */

```

4.11.3 Member Function Documentation

4.11.3.1 __handleKeyPressEvents()

```

void SolarPV::__handleKeyPressEvents (
    void ) [private], [virtual]

```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```

69 {
70     switch (this->event_ptr->key.code) {
71         //...
72
73         default: {
74             // do nothing!
75
76             break;
77         }
78     }
79 }
80
81 return;
82 } /* __handleKeyPressEvents() */

```

4.11.3.2 __handleMouseButtonEvents()

```

void SolarPV::__handleMouseButtonEvents (
    void ) [private], [virtual]

```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```

97 {
98     switch (this->event_ptr->mouseButton.button) {
99         case (sf::Mouse::Left): {
100             //...
101
102             break;
103         }
104
105         case (sf::Mouse::Right): {
106             //...
107
108             break;
109         }
110
111         default: {
112             // do nothing!
113
114             break;
115         }
116     }
117 }
118
119 return;
120 } /* __handleMouseButtonEvents() */

```

4.11.3.3 __setUpTileImprovementSpriteStatic()

```

void SolarPV::__setUpTileImprovementSpriteStatic (
    void ) [private]

```

Helper method to set up tile improvement sprite (static).

```

34 {
35     this->tile_improvement_sprite_static.setTexture(
36         *(this->assets_manager_ptr->getTexture("solar PV array"))
37     );
38
39     this->tile_improvement_sprite_static.setOrigin(
40         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
41         this->tile_improvement_sprite_static.getLocalBounds().height
42     );
43
44     this->tile_improvement_sprite_static.setPosition(
45         this->position_x,

```

```

46         this->position_y - 32
47     );
48
49     this->tile_improvement_sprite_static.setColor(
50         sf::Color(255, 255, 255, 0)
51     );
52
53     return;
54 } /* __setUpTileImprovementSpriteStatic() */

```

4.11.3.4 draw()

```

void SolarPV::draw (
    void ) [virtual]

```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```

254 {
255     // 1. if just built, call base method and return
256     if (this->just_built) {
257         TileImprovement :: draw();
258
259         return;
260     }
261
262     // 1. draw static sprite
263     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
264
265     this->frame++;
266     return;
267 } /* draw() */

```

4.11.3.5 processEvent()

```

void SolarPV::processEvent (
    void ) [virtual]

```

Method to process [SolarPV](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```

209 {
210     if (this->event_ptr->type == sf::Event::KeyPressed) {
211         this->__handleKeyPressEvents();
212     }
213
214     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
215         this->__handleMouseButtonEvents();
216     }
217
218     return;
219 } /* processEvent() */

```

4.11.3.6 processMessage()

```
void SolarPV::processMessage (
    void ) [virtual]
```

Method to process [SolarPV](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
234 {
235     //...
236
237     return;
238 } /* processMessage() */
```

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

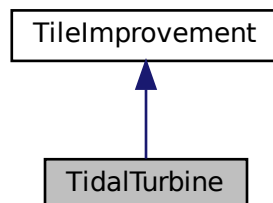
- header/[SolarPV.h](#)
- source/[SolarPV.cpp](#)

4.12 TidalTurbine Class Reference

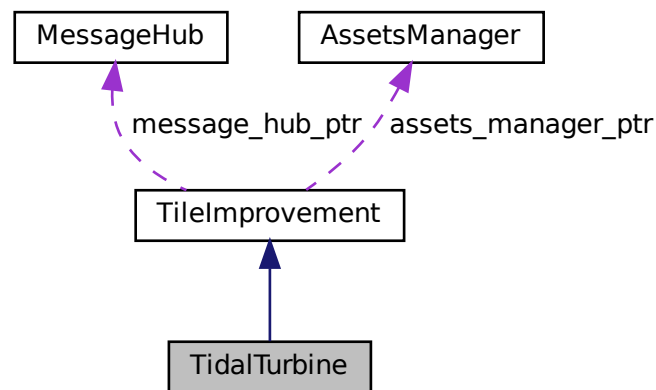
A settlement class (child class of [TileImprovement](#)).

```
#include <TidalTurbine.h>
```

Inheritance diagram for TidalTurbine:



Collaboration diagram for TidalTurbine:



Public Member Functions

- [TidalTurbine](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [TidalTurbine](#) class.
- void [processEvent](#) (void)
Method to process [TidalTurbine](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [TidalTurbine](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~TidalTurbine](#) (void)
Destructor for the [TidalTurbine](#) class.

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.12.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TidalTurbine()

```
TidalTurbine::TidalTurbine (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [TidalTurbine](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
178 :
179 TileImprovement (
180     position_x,
181     position_y,
182     event_ptr,
183     render_window_ptr,
184     assets_manager_ptr,
185     message_hub_ptr
186 )
187 {
188     // 1. set attributes
189
190     // 1.1. private
191     //...
192
193     // 1.2. public
194     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
195
196     this->is_running = false;
197
198     this->tile_improvement_string = "TIDAL TURBINE";
199
200     this->__setUpTileImprovementSpriteAnimated();
201
202     std::cout << "TidalTurbine constructed at " << this << std::endl;
203
204     return;
205 } /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

```
TidalTurbine::~~TidalTurbine (
    void ) [virtual]
```


Destructor for the [TidalTurbine](#) class.

```
306 {
307     std::cout << "TidalTurbine at " << this << " destroyed" << std::endl;
308
309     return;
310 } /* ~TidalTurbine() */
```

4.12.3 Member Function Documentation

4.12.3.1 __handleKeyPressEvents()

```
void TidalTurbine::__handleKeyPressEvents (
    void ) [private], [virtual]
```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```
80 {
81     switch (this->event_ptr->key.code) {
82         //...
83
84         default: {
85             // do nothing!
86
87             break;
88         }
89     }
90 }
91
92 return;
93 } /* __handleKeyPressEvents() */
```

4.12.3.2 __handleMouseButtonEvents()

```
void TidalTurbine::__handleMouseButtonEvents (
    void ) [private], [virtual]
```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
108 {
109     switch (this->event_ptr->mouseButton.button) {
110         case (sf::Mouse::Left): {
111             //...
112
113             break;
114         }
115
116         case (sf::Mouse::Right): {
117             //...
118
119             break;
120         }
121
122         default: {
123             // do nothing!
124
125             break;
126         }
127     }
128 }
129
130 return;
131 } /* __handleMouseButtonEvents() */
```

4.12.3.3 __setUpTileImprovementSpriteAnimated()

```
void TidalTurbine::__setUpTileImprovementSpriteAnimated (
    void ) [private]
```

Helper method to set up tile improvement sprite (static).

```
34 {
35     sf::Sprite diesel_generator_sheet (
36         *(this->assets_manager_ptr->getTexture("tidal turbine"))
37     );
38
39     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
40
41     for (int i = 0; i < n_elements; i++) {
42         this->tile_improvement_sprite_animated.push_back(
43             sf::Sprite(
44                 *(this->assets_manager_ptr->getTexture("tidal turbine")),
45                 sf::IntRect(0, i * 64, 64, 64)
46             )
47         );
48
49         this->tile_improvement_sprite_animated.back().setOrigin(
50             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
51             this->tile_improvement_sprite_animated.back().getLocalBounds().height
52         );
53
54         this->tile_improvement_sprite_animated.back().setPosition(
55             this->position_x,
56             this->position_y - 32
57         );
58
59         this->tile_improvement_sprite_animated.back().setColor(
60             sf::Color(255, 255, 255, 0)
61         );
62     }
63
64     return;
65 } /* __setUpTileImprovementSpriteAnimated() */
```

4.12.3.4 draw()

```
void TidalTurbine::draw (
    void ) [virtual]
```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```
265 {
266     // 1. if just built, call base method and return
267     if (this->just_built) {
268         TileImprovement::draw();
269
270         return;
271     }
272
273
274     // 1. draw first element of animated sprite
275     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
276
277
278     // 2. draw second element of animated sprite
279     if (this->is_running) {
280         //...
281     }
282
283     else {
284         //...
285     }
286
287     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
288
289     this->frame++;
290     return;
291 } /* draw() */
```

4.12.3.5 processEvent()

```
void TidalTurbine::processEvent (
    void ) [virtual]
```

Method to process [TidalTurbine](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```
220 {
221     if (this->event_ptr->type == sf::Event::KeyPressed) {
222         this->__handleKeyPressEvents();
223     }
224
225     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
226         this->__handleMouseButtonEvents();
227     }
228
229     return;
230 } /* processEvent() */
```

4.12.3.6 processMessage()

```
void TidalTurbine::processMessage (
    void ) [virtual]
```

Method to process [TidalTurbine](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
245 {
246     //...
247
248     return;
249 } /* processMessage() */
```

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

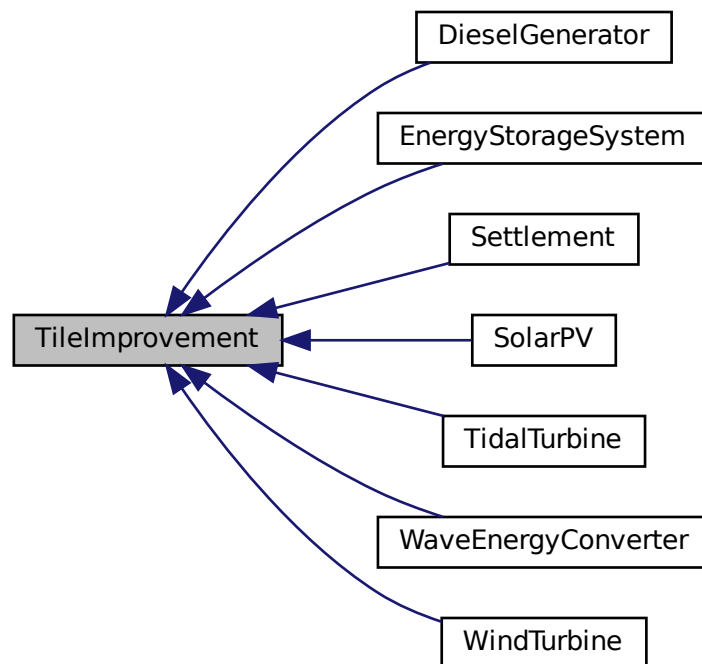
- header/[TidalTurbine.h](#)
- source/[TidalTurbine.cpp](#)

4.13 TileImprovement Class Reference

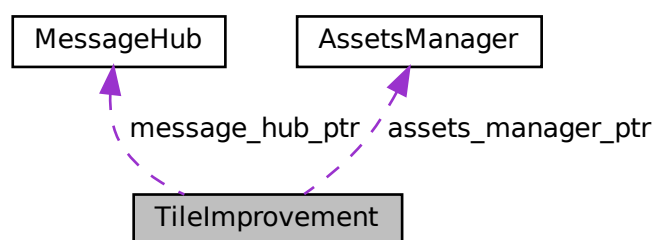
A base class for the tile improvement hierarchy.

```
#include <TileImprovement.h>
```

Inheritance diagram for TileImprovement:



Collaboration diagram for TileImprovement:



Public Member Functions

- **TileImprovement** (double, double, sf::Event *, sf::RenderWindow *, **AssetsManager** *, **MessageHub** *)
*Constructor for the **TileImprovement** class.*
- virtual void **processEvent** (void)
*Method to process **TileImprovement**. To be called once per event.*

- virtual void `processMessage` (void)
Method to process `TileImprovement`. To be called once per message.
- virtual void `draw` (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual `~TileImprovement` (void)
Destructor for the `TileImprovement` class.

Public Attributes

- `TileImprovementType` `tile_improvement_type`
The type of the tile improvement.
- bool `is_running`
A boolean which indicates whether or not the improvement is running.
- bool `is_selected`
A boolean which indicates whether or not the tile is selected.
- bool `just_built`
A boolean which indicates that the improvement was just built.
- int `frame`
The current frame of this object.
- int `credits`
The current balance of credits.
- double `position_x`
The x position of the tile improvement.
- double `position_y`
The y position of the tile improvement.
- std::string `game_phase`
The current phase of the game.
- std::string `tile_improvement_string`
A string representation of the tile improvement type.
- sf::Sprite `tile_improvement_sprite_static`
A static sprite, for decorating the tile.
- std::vector< sf::Sprite > `tile_improvement_sprite_animated`
An animated sprite, for the `ContextMenu` visual screen.

Protected Member Functions

- virtual void `__handleKeyPressEvents` (void)
Helper method to handle key press events.
- virtual void `__handleMouseButtonEvents` (void)
Helper method to handle mouse button events.

Protected Attributes

- sf::Event * `event_ptr`
A pointer to the event class.
- sf::RenderWindow * `render_window_ptr`
A pointer to the render window.
- `AssetsManager` * `assets_manager_ptr`
A pointer to the assets manager.
- `MessageHub` * `message_hub_ptr`
A pointer to the message hub.

4.13.1 Detailed Description

A base class for the tile improvement hierarchy.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TileImprovement()

```
TileImprovement::TileImprovement (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [TileImprovement](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
133 {
134     // 1. set attributes
135
136     // 1.1. protected
137     this->event_ptr = event_ptr;
138     this->render_window_ptr = render_window_ptr;
139
140     this->assets_manager_ptr = assets_manager_ptr;
141     this->message_hub_ptr = message_hub_ptr;
142
143     // 1.2. public
144     this->is_selected = true;
145     this->just_built = true;
146
147     this->frame = 0;
148     this->credits = 0;
149
150     this->position_x = position_x;
151     this->position_y = position_y;
152
153     this->game_phase = "build settlement";
154
155     std::cout << "TileImprovement constructed at " << this << std::endl;
156
157     return;
158 } /* TileImprovement() */
```

4.13.2.2 ~TileImprovement()

```
TileImprovement::~~TileImprovement (
    void ) [virtual]
```

Destructor for the [TileImprovement](#) class.

```
347 {
348     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
349
350     return;
351 } /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __handleKeyPressEvents()

```
void TileImprovement::__handleKeyPressEvents (
    void ) [protected], [virtual]
```

Helper method to handle key press events.

Reimplemented in [WindTurbine](#), [WaveEnergyConverter](#), [TidalTurbine](#), [SolarPV](#), [Settlement](#), [EnergyStorageSystem](#), and [DieselGenerator](#).

```
34 {
35     switch (this->event_ptr->key.code) {
36         //...
37
38
39         default: {
40             // do nothing!
41
42             break;
43         }
44     }
45
46     return;
47 } /* __handleKeyPressEvents() */
```

4.13.3.2 __handleMouseButtonEvents()

```
void TileImprovement::__handleMouseButtonEvents (
    void ) [protected], [virtual]
```

Helper method to handle mouse button events.

Reimplemented in [WindTurbine](#), [WaveEnergyConverter](#), [TidalTurbine](#), [SolarPV](#), [Settlement](#), [EnergyStorageSystem](#), and [DieselGenerator](#).

```
62 {
63     switch (this->event_ptr->mouseButton.button) {
64         case (sf::Mouse::Left): {
65             //...
66
67             break;
68         }
69
70
71         case (sf::Mouse::Right): {
72             //...
73
```

```

74         break;
75     }
76
77
78     default: {
79         // do nothing!
80
81         break;
82     }
83 }
84
85 return;
86 } /* __handleMouseButtonEvents() */

```

4.13.3.3 draw()

```

void TileImprovement::draw (
    void ) [virtual]

```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented in [WindTurbine](#), [WaveEnergyConverter](#), [TidalTurbine](#), [SolarPV](#), [Settlement](#), [EnergyStorageSystem](#), and [DieselGenerator](#).

```

218 {
219     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
220         int alpha = this->tile_improvement_sprite_static.getColor().a;
221
222         alpha += 0.04 * FRAMES_PER_SECOND;
223
224         this->tile_improvement_sprite_static.setColor(
225             sf::Color(255, 255, 255, alpha)
226         );
227
228         this->tile_improvement_sprite_static.move(0, 25 * SECONDS_PER_FRAME);
229
230         if (
231             (alpha >= 255) or
232             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
233         ) {
234             this->tile_improvement_sprite_static.setColor(
235                 sf::Color(255, 255, 255, 255)
236             );
237
238             this->tile_improvement_sprite_static.setPosition(
239                 this->position_x,
240                 this->position_y + 12
241             );
242
243             this->just_built = false;
244             this->assets_manager_ptr->getSound("place_improvement")->play();
245         }
246
247         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
248     }
249
250     else {
251         int alpha = 0;
252
253         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
254             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
255
256             alpha += 0.04 * FRAMES_PER_SECOND;
257
258             this->tile_improvement_sprite_animated[i].setColor(
259                 sf::Color(255, 255, 255, alpha)
260             );
261
262             this->tile_improvement_sprite_animated[i].move(0, 25 * SECONDS_PER_FRAME);
263
264             if (
265                 (alpha >= 255) or
266                 (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
267             ) {
268                 this->tile_improvement_sprite_animated[i].setColor(
269                     sf::Color(255, 255, 255, 255)
270                 );
271             }

```



```

272
273         this->tile_improvement_sprite_animated[i].setPosition(
274             this->position_x,
275             this->position_y + 12
276         );
277     }
278
279     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
280 }
281
282 if (
283     (alpha >= 255) or
284     (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
285 ) {
286     this->just_built = false;
287     this->assets_manager_ptr->getSound("place improvement")->play();
288
289     switch (this->tile_improvement_type) {
290     case (TileImprovementType :: WIND_TURBINE): {
291         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
292             this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
293             this->tile_improvement_sprite_animated[i].move(0, -32);
294         }
295
296         break;
297     }
298
299     case (TileImprovementType :: TIDAL_TURBINE): {
300         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
301             this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
302             this->tile_improvement_sprite_animated[i].move(0, -19);
303         }
304
305         break;
306     }
307
308     case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
309         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
310             this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
311             this->tile_improvement_sprite_animated[i].move(0, -32);
312         }
313
314         break;
315     }
316
317     default: {
318         // do nothing!
319
320         break;
321     }
322 }
323
324 }
325
326 }
327
328 }
329
330 this->frame++;
331 return;
332 } /* draw() */

```

4.13.3.4 processEvent()

```

void TileImprovement::processEvent (
    void ) [virtual]

```

Method to process [TileImprovement](#). To be called once per event.

Reimplemented in [WindTurbine](#), [WaveEnergyConverter](#), [TidalTurbine](#), [SolarPV](#), [Settlement](#), [EnergyStorageSystem](#), and [DieselGenerator](#).

```

173 {
174     if (this->event_ptr->type == sf::Event::KeyPressed) {
175         this->__handleKeyPressEvents();
176     }
177
178     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {

```

```
179         this->__handleMouseButtonEvents();
180     }
181
182     return;
183 } /* processEvent() */
```

4.13.3.5 processMessage()

```
void TileImprovement::processMessage (
    void ) [virtual]
```

Method to process [TileImprovement](#). To be called once per message.

Reimplemented in [WindTurbine](#), [WaveEnergyConverter](#), [TidalTurbine](#), [SolarPV](#), [Settlement](#), [EnergyStorageSystem](#), and [DieselGenerator](#).

```
198 {
199     //...
200
201     return;
202 } /* processMessage() */
```

4.13.4 Member Data Documentation

4.13.4.1 assets_manager_ptr

```
AssetsManager* TileImprovement::assets_manager_ptr [protected]
```

A pointer to the assets manager.

4.13.4.2 credits

```
int TileImprovement::credits
```

The current balance of credits.

4.13.4.3 event_ptr

```
sf::Event* TileImprovement::event_ptr [protected]
```

A pointer to the event class.

4.13.4.4 frame

```
int TileImprovement::frame
```

The current frame of this object.

4.13.4.5 game_phase

```
std::string TileImprovement::game_phase
```

The current phase of the game.

4.13.4.6 is_running

```
bool TileImprovement::is_running
```

A boolean which indicates whether or not the improvement is running.

4.13.4.7 is_selected

```
bool TileImprovement::is_selected
```

A boolean which indicates whether or not the tile is selected.

4.13.4.8 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.9 message_hub_ptr

```
MessageHub* TileImprovement::message_hub_ptr [protected]
```

A pointer to the message hub.

4.13.4.10 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.11 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.12 render_window_ptr

```
sf::RenderWindow* TileImprovement::render_window_ptr [protected]
```

A pointer to the render window.

4.13.4.13 tile_improvement_sprite_animated

```
std::vector<sf::Sprite> TileImprovement::tile_improvement_sprite_animated
```

An animated sprite, for the [ContextMenu](#) visual screen.

4.13.4.14 tile_improvement_sprite_static

```
sf::Sprite TileImprovement::tile_improvement_sprite_static
```

A static sprite, for decorating the tile.

4.13.4.15 tile_improvement_string

```
std::string TileImprovement::tile_improvement_string
```

A string representation of the tile improvement type.

4.13.4.16 tile_improvement_type

`TileImprovementType` `TileImprovement::tile_improvement_type`

The type of the tile improvement.

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

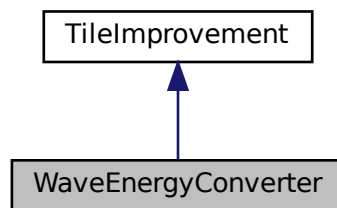
- header/[TileImprovement.h](#)
- source/[TileImprovement.cpp](#)

4.14 WaveEnergyConverter Class Reference

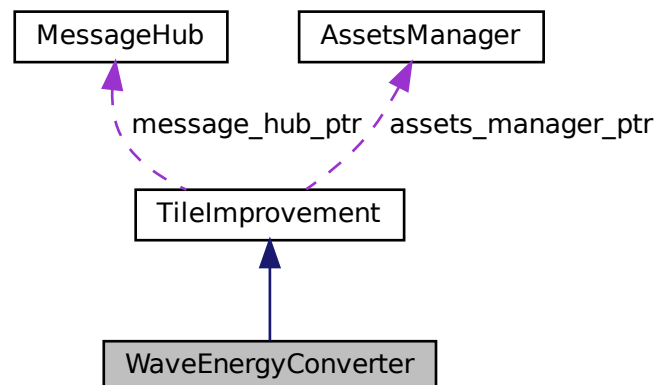
A settlement class (child class of [TileImprovement](#)).

```
#include <WaveEnergyConverter.h>
```

Inheritance diagram for WaveEnergyConverter:



Collaboration diagram for WaveEnergyConverter:



Public Member Functions

- [WaveEnergyConverter](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [WaveEnergyConverter](#) class.
- void [processEvent](#) (void)
Method to process [WaveEnergyConverter](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [WaveEnergyConverter](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~WaveEnergyConverter](#) (void)
Destructor for the [WaveEnergyConverter](#) class.

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.14.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.14.2 Constructor & Destructor Documentation

4.14.2.1 WaveEnergyConverter()

```
WaveEnergyConverter::WaveEnergyConverter (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [WaveEnergyConverter](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```

178 :
179 TileImprovement (
180     position_x,
181     position_y,
182     event_ptr,
183     render_window_ptr,
184     assets_manager_ptr,
185     message_hub_ptr
186 )
187 {
188     // 1. set attributes
189
190     // 1.1. private
191     //...
192
193     // 1.2. public
194     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
195
196     this->is_running = false;
197
198     this->tile_improvement_string = "WAVE ENERGY";
199
200     this->__setUpTileImprovementSpriteAnimated();
201
202     std::cout << "WaveEnergyConverter constructed at " << this << std::endl;
203
204     return;
205 } /* WaveEnergyConverter() */

```

4.14.2.2 ~WaveEnergyConverter()

```

WaveEnergyConverter::~WaveEnergyConverter (
    void ) [virtual]

```

Destructor for the [WaveEnergyConverter](#) class.

```

306 {
307     std::cout << "WaveEnergyConverter at " << this << " destroyed" << std::endl;
308
309     return;
310 } /* ~WaveEnergyConverter() */

```

4.14.3 Member Function Documentation

4.14.3.1 __handleKeyPressEvents()

```

void WaveEnergyConverter::__handleKeyPressEvents (
    void ) [private], [virtual]

```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```

80 {
81     switch (this->event_ptr->key.code) {
82         //...
83
84
85         default: {
86             // do nothing!
87
88             break;
89         }
90     }
91
92     return;
93 } /* __handleKeyPressEvents() */

```

4.14.3.2 __handleMouseButtonEvents()

```

void WaveEnergyConverter::__handleMouseButtonEvents (
    void ) [private], [virtual]

```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```

108 {
109     switch (this->event_ptr->mouseButton.button) {
110         case (sf::Mouse::Left): {
111             //...
112
113             break;
114         }
115
116
117         case (sf::Mouse::Right): {
118             //...
119
120             break;
121         }
122
123
124         default: {
125             // do nothing!
126
127             break;
128         }
129     }
130
131     return;
132 } /* __handleMouseButtonEvents() */

```

4.14.3.3 __setUpTileImprovementSpriteAnimated()

```

void WaveEnergyConverter::__setUpTileImprovementSpriteAnimated (
    void ) [private]

```

Helper method to set up tile improvement sprite (static).

```

34 {
35     sf::Sprite diesel_generator_sheet (
36         *(this->assets_manager_ptr->getTexture("wave energy converter"))
37     );
38
39     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
40
41     for (int i = 0; i < n_elements; i++) {
42         this->tile_improvement_sprite_animated.push_back(
43             sf::Sprite(
44                 *(this->assets_manager_ptr->getTexture("wave energy converter")),
45                 sf::IntRect(0, i * 64, 64, 64)

```



```

46         )
47     };
48
49     this->tile_improvement_sprite_animated.back().setOrigin(
50         this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
51         this->tile_improvement_sprite_animated.back().getLocalBounds().height
52     );
53
54     this->tile_improvement_sprite_animated.back().setPosition(
55         this->position_x,
56         this->position_y - 32
57     );
58
59     this->tile_improvement_sprite_animated.back().setColor(
60         sf::Color(255, 255, 255, 0)
61     );
62 }
63
64 return;
65 } /* __setUpTileImprovementSpriteAnimated() */

```

4.14.3.4 draw()

```

void WaveEnergyConverter::draw (
    void ) [virtual]

```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```

265 {
266     // 1. if just built, call base method and return
267     if (this->just_built) {
268         TileImprovement::draw();
269
270         return;
271     }
272
273
274     // 1. draw first element of animated sprite
275     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
276
277
278     // 2. draw second element of animated sprite
279     if (this->is_running) {
280         //...
281     }
282
283     else {
284         //...
285     }
286
287     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
288
289     this->frame++;
290     return;
291 } /* draw() */

```

4.14.3.5 processEvent()

```

void WaveEnergyConverter::processEvent (
    void ) [virtual]

```

Method to process [WaveEnergyConverter](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```

220 {
221     if (this->event_ptr->type == sf::Event::KeyPressed) {

```

```
222     this->__handleKeyPressEvents();
223 }
224
225 if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
226     this->__handleMouseButtonEvents();
227 }
228
229 return;
230 } /* processEvent() */
```

4.14.3.6 processMessage()

```
void WaveEnergyConverter::processMessage (
    void ) [virtual]
```

Method to process [WaveEnergyConverter](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
245 {
246     //...
247
248     return;
249 } /* processMessage() */
```

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

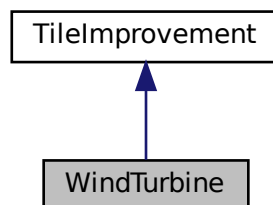
- header/[WaveEnergyConverter.h](#)
- source/[WaveEnergyConverter.cpp](#)

4.15 WindTurbine Class Reference

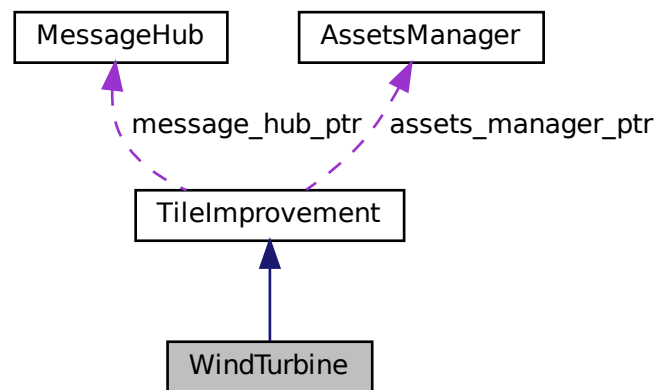
A settlement class (child class of [TileImprovement](#)).

```
#include <WindTurbine.h>
```

Inheritance diagram for WindTurbine:



Collaboration diagram for WindTurbine:



Public Member Functions

- [WindTurbine](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [WindTurbine](#) class.
- void [processEvent](#) (void)
Method to process [WindTurbine](#). To be called once per event.
- void [processMessage](#) (void)
Method to process [WindTurbine](#). To be called once per message.
- void [draw](#) (void)
Method to draw the hex tile to the render window. To be called once per frame.
- virtual [~WindTurbine](#) (void)
Destructor for the [WindTurbine](#) class.

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.

Additional Inherited Members

4.15.1 Detailed Description

A settlement class (child class of [TileImprovement](#)).

4.15.2 Constructor & Destructor Documentation

4.15.2.1 WindTurbine()

```
WindTurbine::WindTurbine (
    double position_x,
    double position_y,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [WindTurbine](#) class.

Ref: [Wikipedia \[2023\]](#)

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>event_ptr</i>	Pointer to the event class.
<i>render_window_ptr</i>	Pointer to the render window.
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>message_hub_ptr</i>	Pointer to the message hub.

```
178 :
179 TileImprovement (
180     position_x,
181     position_y,
182     event_ptr,
183     render_window_ptr,
184     assets_manager_ptr,
185     message_hub_ptr
186 )
187 {
188     // 1. set attributes
189
190     // 1.1. private
191     //...
192
193     // 1.2. public
194     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
195
196     this->is_running = false;
197
198     this->tile_improvement_string = "WIND TURBINE";
199
200     this->__setUpTileImprovementSpriteAnimated();
201
202     std::cout << "WindTurbine constructed at " << this << std::endl;
203
204     return;
205 } /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

```
WindTurbine::~~WindTurbine (
    void ) [virtual]
```

Destructor for the [WindTurbine](#) class.

```
306 {
307     std::cout << "WindTurbine at " << this << " destroyed" << std::endl;
308
309     return;
310 } /* ~WindTurbine() */
```

4.15.3 Member Function Documentation

4.15.3.1 __handleKeyPressEvents()

```
void WindTurbine::__handleKeyPressEvents (
    void ) [private], [virtual]
```

Helper method to handle key press events.

Reimplemented from [TileImprovement](#).

```
80 {
81     switch (this->event_ptr->key.code) {
82         //...
83
84         default: {
85             // do nothing!
86
87             break;
88         }
89     }
90 }
91
92 return;
93 } /* __handleKeyPressEvents() */
```

4.15.3.2 __handleMouseButtonEvents()

```
void WindTurbine::__handleMouseButtonEvents (
    void ) [private], [virtual]
```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
108 {
109     switch (this->event_ptr->mouseButton.button) {
110         case (sf::Mouse::Left): {
111             //...
112
113             break;
114         }
115
116         case (sf::Mouse::Right): {
117             //...
118
119             break;
120         }
121     }
122
123     default: {
124         // do nothing!
125
126         break;
127     }
128 }
129
130
131 return;
132 } /* __handleMouseButtonEvents() */
```

4.15.3.3 __setUpTileImprovementSpriteAnimated()

```
void WindTurbine::__setUpTileImprovementSpriteAnimated (
    void ) [private]
```

Helper method to set up tile improvement sprite (static).

```
34 {
35     sf::Sprite diesel_generator_sheet (
36         *(this->assets_manager_ptr->getTexture("wind turbine"))
37     );
38
39     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
40
41     for (int i = 0; i < n_elements; i++) {
42         this->tile_improvement_sprite_animated.push_back(
43             sf::Sprite(
44                 *(this->assets_manager_ptr->getTexture("wind turbine")),
45                 sf::IntRect(0, i * 64, 64, 64)
46             )
47         );
48
49         this->tile_improvement_sprite_animated.back().setOrigin(
50             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
51             this->tile_improvement_sprite_animated.back().getLocalBounds().height
52         );
53
54         this->tile_improvement_sprite_animated.back().setPosition(
55             this->position_x,
56             this->position_y - 32
57         );
58
59         this->tile_improvement_sprite_animated.back().setColor(
60             sf::Color(255, 255, 255, 0)
61         );
62     }
63
64     return;
65 } /* __setUpTileImprovementSpriteAnimated() */
```

4.15.3.4 draw()

```
void WindTurbine::draw (
    void ) [virtual]
```

Method to draw the hex tile to the render window. To be called once per frame.

Reimplemented from [TileImprovement](#).

```
265 {
266     // 1. if just built, call base method and return
267     if (this->just_built) {
268         TileImprovement::draw();
269
270         return;
271     }
272
273
274     // 1. draw first element of animated sprite
275     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
276
277
278     // 2. draw second element of animated sprite
279     if (this->is_running) {
280         //...
281     }
282
283     else {
284         //...
285     }
286
287     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
288
289     this->frame++;
290     return;
291 } /* draw() */
```

4.15.3.5 processEvent()

```
void WindTurbine::processEvent (
    void ) [virtual]
```

Method to process [WindTurbine](#). To be called once per event.

Reimplemented from [TileImprovement](#).

```
220 {
221     if (this->event_ptr->type == sf::Event::KeyPressed) {
222         this->__handleKeyPressEvents();
223     }
224
225     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
226         this->__handleMouseButtonEvents();
227     }
228
229     return;
230 } /* processEvent() */
```

4.15.3.6 processMessage()

```
void WindTurbine::processMessage (
    void ) [virtual]
```

Method to process [WindTurbine](#). To be called once per message.

Reimplemented from [TileImprovement](#).

```
245 {
246     //...
247
248     return;
249 } /* processMessage() */
```

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

- header/[WindTurbine.h](#)
- source/[WindTurbine.cpp](#)

Chapter 5

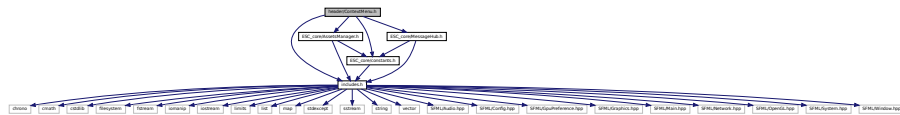
File Documentation

5.1 header/ContextMenu.h File Reference

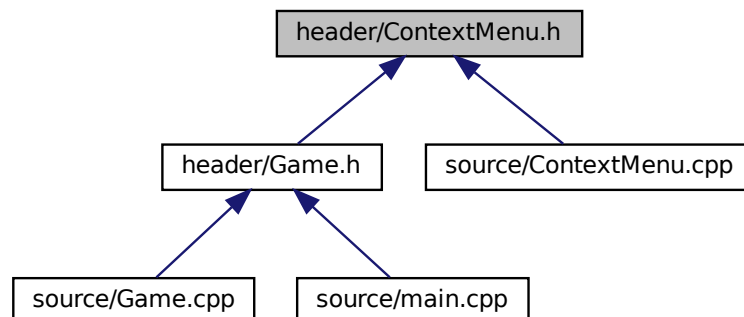
Header file for the [ContextMenu](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

Include dependency graph for ContextMenu.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [ContextMenu](#)

A class which defines a context menu for the game.

Enumerations

- enum [ConsoleState](#) {
[NONE_STATE](#) , [READY](#) , [MENU](#) , [TILE](#) ,
[N_CONSOLE_STATES](#) }

An enumeration of the different console screen states.

5.1.1 Detailed Description

Header file for the [ContextMenu](#) class.

5.1.2 Enumeration Type Documentation

5.1.2.1 ConsoleState

enum [ConsoleState](#)

An enumeration of the different console screen states.

Enumerator

NONE_STATE	None state (for initialization)
READY	Ready (default) state.
MENU	Game menu state.
TILE	Tile context state.
N_CONSOLE_STATES	A simple hack to get the number of console screen states.

```

34         {
35     NONE_STATE,
36     READY,
37     MENU,
38     TILE,
39     N_CONSOLE_STATES
40 };

```

5.2 header/DieselGenerator.h File Reference

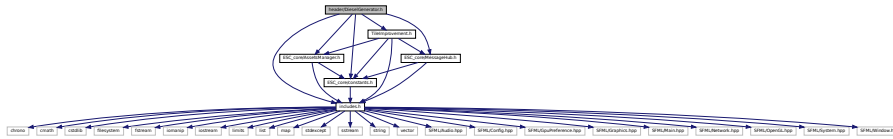
Header file for the [DieselGenerator](#) class.

```

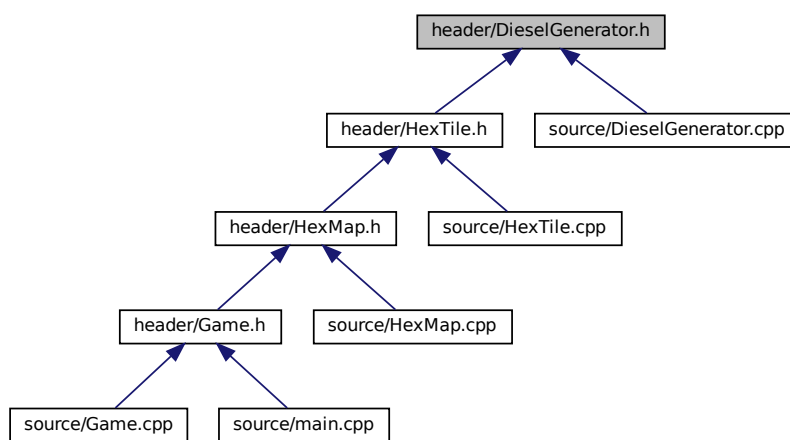
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"

```

```
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
Include dependency graph for DieselGenerator.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [DieselGenerator](#)
A settlement class (child class of [TileImprovement](#)).

5.2.1 Detailed Description

Header file for the [DieselGenerator](#) class.

5.3 header/EnergyStorageSystem.h File Reference

Header file for the [EnergyStorageSystem](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```


- class `AssetsManager`
A class which manages visual and sound assets.

Header file for the `AssetsManager` class.

Header file for various constants.

Functions

- `const sf::Color FOREST_GREEN` (34, 139, 34)
The base colour of a forest tile.
- `const sf::Color LAKE_BLUE` (0, 102, 204)
The base colour of a lake (water) tile.
- `const sf::Color MOUNTAINS_GREY` (97, 110, 113)
The base colour of a mountains tile.
- `const sf::Color OCEAN_BLUE` (0, 51, 102)
The base colour of an ocean (water) tile.
- `const sf::Color PLAINS_YELLOW` (245, 222, 133)
The base colour of a plains tile.
- `const sf::Color RESOURCE_CHIP_GREY` (175, 175, 175, 250)
The base colour of the resource chip (backing).
- `const sf::Color MENU_FRAME_GREY` (185, 187, 182)
The base colour of the context menu frame.
- `const sf::Color MONOCHROME_SCREEN_BACKGROUND` (40, 40, 40)
The base colour of old monochrome screens.
- `const sf::Color VISUAL_SCREEN_FRAME_GREY` (151, 151, 143)
The base colour of the framing of the visual screen.
- `const sf::Color MONOCHROME_TEXT_GREEN` (0, 255, 102)
The base colour of old monochrome text (green).
- `const sf::Color MONOCHROME_TEXT_AMBER` (255, 176, 0)
The base colour of old monochrome text (amber).
- `const sf::Color MONOCHROME_TEXT_RED` (255, 44, 0)
The base colour of old monochrome text (red).

Variables

- `const double FLOAT_TOLERANCE` = 1e-6
Tolerance for floating point equality tests.
- `const unsigned long long int SECONDS_PER_YEAR` = 31537970
- `const unsigned long long int SECONDS_PER_MONTH` = 2628164
- `const int FRAMES_PER_SECOND` = 60
Target frames per second.
- `const double SECONDS_PER_FRAME` = 1.0 / 60
Target seconds per frame (just reciprocal of target frames per second).
- `const int GAME_WIDTH` = 1200
Width of the game space.
- `const int GAME_HEIGHT` = 800
Height of the game space.
- `const std::vector< double > TILE_TYPE_CUMULATIVE_PROBABILITIES`
Cumulative probabilities for each tile type (to support procedural generation).
- `const std::vector< double > TILE_RESOURCE_CUMULATIVE_PROBABILITIES`
Cumulative probabilities for each tile resource (to support procedural generation).
- `const std::string TILE_SELECTED_CHANNEL` = "TILE SELECTED CHANNEL"
A message channel for tile selection messages.
- `const std::string NO_TILE_SELECTED_CHANNEL` = "NO TILE SELECTED CHANNEL"
A message channel for no tile selected messages.
- `const std::string TILE_STATE_CHANNEL` = "TILE STATE CHANNEL"

- A message channel for tile state messages.*
- const std::string `HEX_MAP_CHANNEL` = "HEX MAP CHANNEL"
- A message channel for hex map messages.*
- const int `CLEAR_FOREST_COST` = 40
- The cost of clearing a forest tile.*
- const int `CLEAR_MOUNTAINS_COST` = 250
- The cost of clearing a mountains tile.*
- const int `CLEAR_PLAINS_COST` = 20
- The cost of clearing a plains tile.*
- const int `DIESEL_GENERATOR_BUILD_COST` = 100
- The cost of building (or upgrading) a diesel generator.*
- const int `WIND_TURBINE_BUILD_COST` = 400
- The cost of building (or upgrading) a wind turbine.*
- const double `WIND_TURBINE_WATER_BUILD_MULTIPLIER` = 1.25
- The additional cost of building on water.*
- const int `SOLAR_PV_BUILD_COST` = 300
- The cost of building (or upgrading) a solar PV array.*
- const double `SOLAR_PV_WATER_BUILD_MULTIPLIER` = 1.5
- The additional cost of building on water.*
- const int `TIDAL_TURBINE_BUILD_COST` = 600
- The cost of building (or upgrading) a tidal turbine.*
- const int `WAVE_ENERGY_CONVERTER_BUILD_COST` = 800
- The cost of building (or upgrading) a wave energy converter.*
- const int `ENERGY_STORAGE_SYSTEM_BUILD_COST` = 400
- The cost of building (or upgrading) an energy storage system.*
- const int `STARTING_CREDITS` = 99999
- The starting balance of credits.*
- const int `EMISSIONS_LIFETIME_LIMIT_TONNES` = 1500
- The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.*
- const int `RESOURCE_ASSESSMENT_COST` = 20
- The cost of doing a resource assessment.*
- const int `BUILD_SETTLEMENT_COST` = 250
- The cost of building a settlement.*
- const int `STARTING_POPULATION` = 100
- The starting population of a settlement.*
- const double `CO2E_KG_PER_LITRE_DIESEL` = 3.1596
- The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.*
- const std::string `GAME_CHANNEL` = "GAME CHANNEL"
- A message channel for game messages.*
- const std::string `GAME_STATE_CHANNEL` = "GAME STATE CHANNEL"
- A message channel for game state messages.*

5.5.1 Detailed Description

Header file for various constants.

5.5.2 Function Documentation

5.5.2.1 FOREST_GREEN()

```
const sf::Color FOREST_GREEN (
    34 ,
    139 ,
    34 )
```

The base colour of a forest tile.

5.5.2.2 LAKE_BLUE()

```
const sf::Color LAKE_BLUE (
    0 ,
    102 ,
    204 )
```

The base colour of a lake (water) tile.

5.5.2.3 MENU_FRAME_GREY()

```
const sf::Color MENU_FRAME_GREY (
    185 ,
    187 ,
    182 )
```

The base colour of the context menu frame.

5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()

```
const sf::Color MONOCHROME_SCREEN_BACKGROUND (
    40 ,
    40 ,
    40 )
```

The base colour of old monochrome screens.

5.5.2.5 MONOCHROME_TEXT_AMBER()

```
const sf::Color MONOCHROME_TEXT_AMBER (
    255 ,
    176 ,
    0 )
```

The base colour of old monochrome text (amber).

5.5.2.6 MONOCHROME_TEXT_GREEN()

```
const sf::Color MONOCHROME_TEXT_GREEN (
    0 ,
    255 ,
    102 )
```

The base colour of old monochrome text (green).

5.5.2.7 MONOCHROME_TEXT_RED()

```
const sf::Color MONOCHROME_TEXT_RED (
    255 ,
    44 ,
    0 )
```

The base colour of old monochrome text (red).

5.5.2.8 MOUNTAINS_GREY()

```
const sf::Color MOUNTAINS_GREY (
    97 ,
    110 ,
    113 )
```

The base colour of a mountains tile.

5.5.2.9 OCEAN_BLUE()

```
const sf::Color OCEAN_BLUE (
    0 ,
    51 ,
    102 )
```

The base colour of an ocean (water) tile.

5.5.2.10 PLAINS_YELLOW()

```
const sf::Color PLAINS_YELLOW (
    245 ,
    222 ,
    133 )
```

The base colour of a plains tile.

5.5.2.11 RESOURCE_CHIP_GREY()

```
const sf::Color RESOURCE_CHIP_GREY (
    175 ,
    175 ,
    175 ,
    250 )
```

The base colour of the resource chip (backing).

5.5.2.12 VISUAL_SCREEN_FRAME_GREY()

```
const sf::Color VISUAL_SCREEN_FRAME_GREY (
    151 ,
    151 ,
    143 )
```

The base colour of the framing of the visual screen.

5.5.3 Variable Documentation

5.5.3.1 BUILD_SETTLEMENT_COST

```
const int BUILD_SETTLEMENT_COST = 250
```

The cost of building a settlement.

5.5.3.2 CLEAR_FOREST_COST

```
const int CLEAR_FOREST_COST = 40
```

The cost of clearing a forest tile.

5.5.3.3 CLEAR_MOUNTAINS_COST

```
const int CLEAR_MOUNTAINS_COST = 250
```

The cost of clearing a mountains tile.

5.5.3.4 CLEAR_PLAINS_COST

```
const int CLEAR_PLAINS_COST = 20
```

The cost of clearing a plains tile.

5.5.3.5 CO2E_KG_PER_LITRE_DIESEL

```
const double CO2E_KG_PER_LITRE_DIESEL = 3.1596
```

The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.

5.5.3.6 DIESEL_GENERATOR_BUILD_COST

```
const int DIESEL_GENERATOR_BUILD_COST = 100
```

The cost of building (or upgrading) a diesel generator.

5.5.3.7 EMISSIONS_LIFETIME_LIMIT_TONNES

```
const int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500
```

The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.

5.5.3.8 ENERGY_STORAGE_SYSTEM_BUILD_COST

```
const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 400
```

The cost of building (or upgrading) an energy storage system.

5.5.3.9 FLOAT_TOLERANCE

```
const double FLOAT_TOLERANCE = 1e-6
```

Tolerance for floating point equality tests.

5.5.3.10 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

5.5.3.11 GAME_CHANNEL

```
const std::string GAME_CHANNEL = "GAME CHANNEL"
```

A message channel for game messages.

5.5.3.12 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

5.5.3.13 GAME_STATE_CHANNEL

```
const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"
```

A message channel for game state messages.

5.5.3.14 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.15 HEX_MAP_CHANNEL

```
const std::string HEX_MAP_CHANNEL = "HEX MAP CHANNEL"
```

A message channel for hex map messages.

5.5.3.16 NO_TILE_SELECTED_CHANNEL

```
const std::string NO_TILE_SELECTED_CHANNEL = "NO TILE SELECTED CHANNEL"
```

A message channel for no tile selected messages.

5.5.3.17 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.18 SECONDS_PER_FRAME

```
const double SECONDS_PER_FRAME = 1.0 / 60
```

Target seconds per frame (just reciprocal of target frames per second).

5.5.3.19 SECONDS_PER_MONTH

```
const unsigned long long int SECONDS_PER_MONTH = 2628164
```

5.5.3.20 SECONDS_PER_YEAR

```
const unsigned long long int SECONDS_PER_YEAR = 31537970
```

5.5.3.21 SOLAR_PV_BUILD_COST

```
const int SOLAR_PV_BUILD_COST = 300
```

The cost of building (or upgrading) a solar PV array.

5.5.3.22 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.23 STARTING_CREDITS

```
const int STARTING_CREDITS = 99999
```

The starting balance of credits.

5.5.3.24 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.25 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

The cost of building (or upgrading) a tidal turbine.

5.5.3.26 TILE_RESOURCE_CUMULATIVE_PROBABILITIES

```
const std::vector<double> TILE_RESOURCE_CUMULATIVE_PROBABILITIES
```

Initial value:

```
= {  
    0.10,  
    0.30,  
    0.70,  
    0.90,  
    1.00  
}
```

Cumulative probabilities for each tile resource (to support procedural generation).

5.5.3.27 TILE_SELECTED_CHANNEL

```
const std::string TILE_SELECTED_CHANNEL = "TILE SELECTED CHANNEL"
```

A message channel for tile selection messages.

5.5.3.28 TILE_STATE_CHANNEL

```
const std::string TILE_STATE_CHANNEL = "TILE STATE CHANNEL"
```

A message channel for tile state messages.

5.5.3.29 TILE_TYPE_CUMULATIVE_PROBABILITIES

```
const std::vector<double> TILE_TYPE_CUMULATIVE_PROBABILITIES
```

Initial value:

```
= {  
    0.25,  
    0.50,  
    0.75,  
    1.00  
}
```

Cumulative probabilities for each tile type (to support procedural generation).

5.5.3.30 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800
```

The cost of building (or upgrading) a wave energy converter.

5.5.3.31 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

The cost of building (or upgrading) a wind turbine.

5.5.3.32 WIND_TURBINE_WATER_BUILD_MULTIPLIER

```
const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25
```

The additional cost of building on water.

5.7.1 Detailed Description

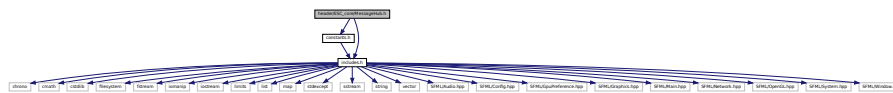
Header file for various includes.

Ref: [Gomila \[2023\]](#)

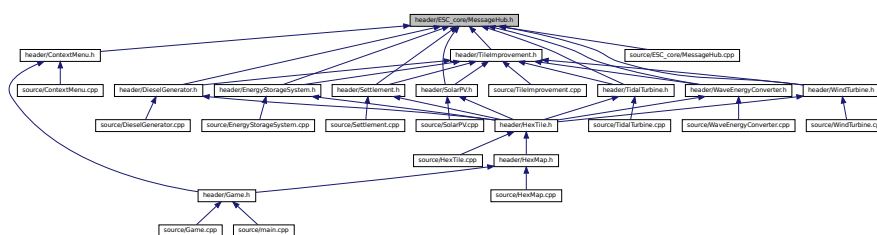
5.8 header/ESC_core/MessageHub.h File Reference

Header file for the [MessageHub](#) class.

```
#include "constants.h"
#include "includes.h"
Include dependency graph for MessageHub.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- struct [Message](#)
A structure which defines a standard message format.
- class [MessageHub](#)
A class which acts as a central hub for inter-object message traffic.

5.8.1 Detailed Description

Header file for the [MessageHub](#) class.

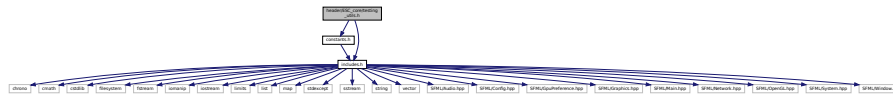
5.9 header/ESC_core/testing_utils.h File Reference

Header file for various testing utilities.

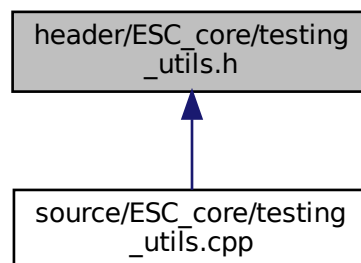
```
#include "constants.h"
```

```
#include "includes.h"
```

Include dependency graph for testing_utils.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [printGreen](#) (std::string)
A function that sends green text to std::cout.
- void [printGold](#) (std::string)
A function that sends gold text to std::cout.
- void [printRed](#) (std::string)
A function that sends red text to std::cout.
- void [testFloatEquals](#) (double, double, std::string, int)
Tests for the equality of two floating point numbers x and y (to within `FLOAT_TOLERANCE`).
- void [testGreaterThan](#) (double, double, std::string, int)
Tests if $x > y$.
- void [testGreaterThanOrEqualTo](#) (double, double, std::string, int)
Tests if $x \geq y$.
- void [testLessThan](#) (double, double, std::string, int)
Tests if $x < y$.
- void [testLessThanOrEqualTo](#) (double, double, std::string, int)
Tests if $x \leq y$.
- void [testTruth](#) (bool, std::string, int)
Tests if the given statement is true.
- void [expectedErrorNotDetected](#) (std::string, int)
A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

5.9.1 Detailed Description

Header file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

5.9.2 Function Documentation

5.9.2.1 expectedErrorNotDetected()

```
void expectedErrorNotDetected (
    std::string file,
    int line )
```

A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

Parameters

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

5.9.2.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {
83     std::cout << "\x1B[33m" << input_str << "\033[0m";
84     return;
85 } /* printGold() */
```

5.9.2.3 printGreen()

```
void printGreen (
    std::string input_str )
```

A function that sends green text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
62 {
63     std::cout << "\x1B[32m" << input_str << "\033[0m";
64     return;
65 } /* printGreen() */
```

5.9.2.4 printRed()

```
void printRed (
    std::string input_str )
```

A function that sends red text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
102 {
103     std::cout << "\x1B[31m" << input_str << "\033[0m";
104     return;
105 } /* printRed() */
```

5.9.2.5 testFloatEquals()

```
void testFloatEquals (
    double x,
    double y,
    std::string file,
    int line )
```

Tests for the equality of two floating point numbers *x* and *y* (to within `FLOAT_TOLERANCE`).

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in " <code>__FILE__</code> ").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in " <code>__LINE__</code> ").

```
136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
```

```

139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";
150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif
156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

5.9.2.6 testGreaterThan()

```

void testGreaterThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x > y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif
207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

5.9.2.7 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    double x,

```

```
double y,
std::string file,
int line )
```

Tests if $x \geq y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);
260     return;
261 } /* testGreaterThanOrEqualTo() */
```

5.9.2.8 testLessThan()

```
void testLessThan (
    double x,
    double y,
    std::string file,
    int line )
```

Tests if $x < y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
```

```

301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);
311     return;
312 } /* testLessThan() */

```

5.9.2.9 testLessThanOrEqualTo()

```

void testLessThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x \leq y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;
363 } /* testLessThanOrEqualTo() */

```

5.9.2.10 testTruth()

```

void testTruth (
    bool statement,
    std::string file,
    int line )

```

Tests if the given statement is true.

Parameters

<i>statement</i>	The statement whose truth is to be tested ("1 == 0", for example).
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\n";
400     error_str += "Given statement is not true";
401
402     #ifdef _WIN32
403         std::cout << error_str << std::endl;
404     #endif
405
406     throw std::runtime_error(error_str);
407     return;
408 } /* testTruth() */

```

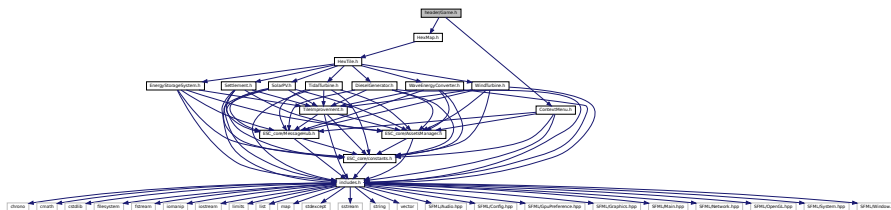
5.10 header/Game.h File Reference

```

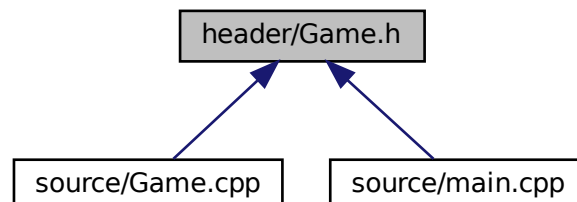
#include "HexMap.h"
#include "ContextMenu.h"

```

Include dependency graph for Game.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Game](#)

A class which acts as the central class for the game, by containing all other classes and implementing the game loop.

Enumerations

- enum [GamePhase](#) {
[BUILD_SETTLEMENT](#) , [SYSTEM_MANAGEMENT](#) , [LOSS_EMISSIONS](#) , [LOSS_DEMAND](#) ,
[LOSS_CREDITS](#) , [VICTORY](#) , [N_GAME_PHASES](#) }

An enumeration of the various game phases.

5.10.1 Enumeration Type Documentation

5.10.1.1 GamePhase

enum [GamePhase](#)

An enumeration of the various game phases.

Enumerator

BUILD_SETTLEMENT	The settlement building phase.
SYSTEM_MANAGEMENT	The system management phase (main phase of play).
LOSS_EMISSIONS	A loss due to excessive emissions.
LOSS_DEMAND	A loss due to failing to meet the demand.
LOSS_CREDITS	A loss due to running out of credits.
VICTORY	A victory (12 consecutive months of zero emissions).
N_GAME_PHASES	A simple hack to get the number of elements in GamePhase.

```

32     {
33         BUILD_SETTLEMENT,
34         SYSTEM_MANAGEMENT,
35         LOSS_EMISSIONS,
36         LOSS_DEMAND,
37         LOSS_CREDITS,
38         VICTORY,
39         N_GAME_PHASES
40 };    /* GamePhase */

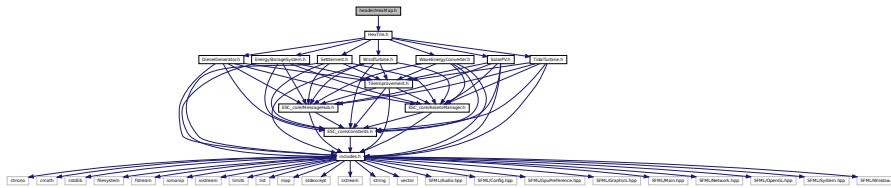
```

5.11 header/HexMap.h File Reference

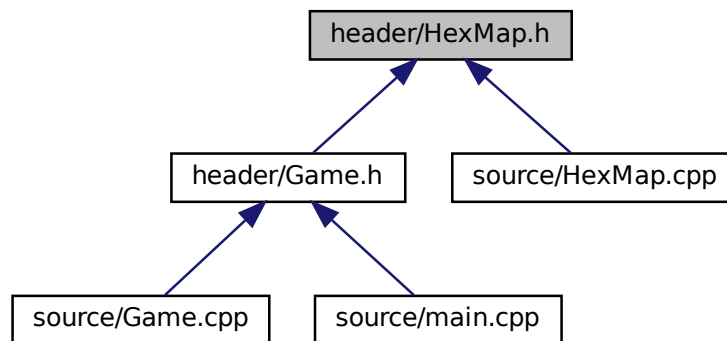
Header file for the [HexMap](#) class.

```
#include "HexTile.h"
```

Include dependency graph for HexMap.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [HexMap](#)
A class which defines a hex map of hex tiles.

5.11.1 Detailed Description

Header file for the [HexMap](#) class.

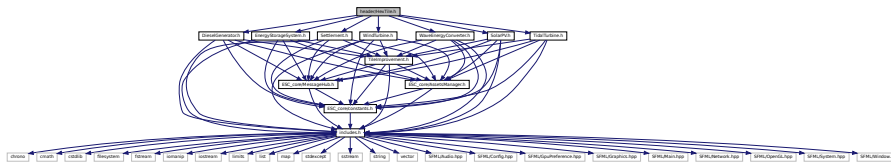
5.12 header/HexTile.h File Reference

Header file for the [Game](#) class.

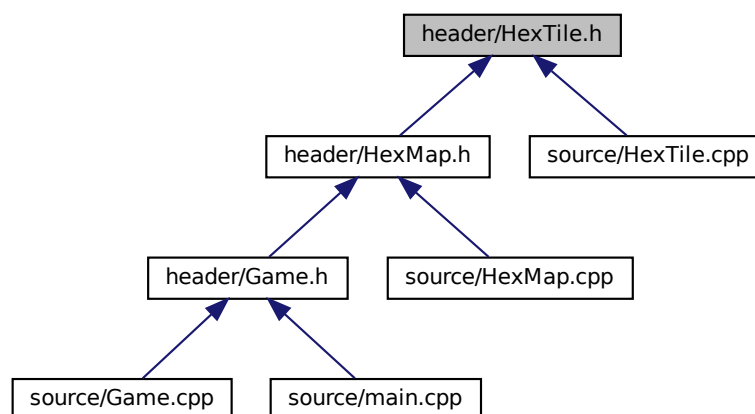
```
#include "DieselGenerator.h"
#include "EnergyStorageSystem.h"
#include "Settlement.h"
#include "SolarPV.h"
#include "TidalTurbine.h"
#include "WaveEnergyConverter.h"
```

```
#include "WindTurbine.h"
```

Include dependency graph for HexTile.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [HexTile](#)
A class which defines a hex tile of the hex map.

Enumerations

- enum [TileType](#) {
 [NONE_TYPE](#) , [FOREST](#) , [LAKE](#) , [MOUNTAINS](#) ,
 [OCEAN](#) , [PLAINS](#) , [N_TILE_TYPES](#) }
An enumeration of the different tile types.
- enum [TileResource](#) {
 [POOR](#) , [BELOW_AVERAGE](#) , [AVERAGE](#) , [ABOVE_AVERAGE](#) ,
 [GOOD](#) , [N_TILE_RESOURCES](#) }
An enumeration of the different tile resource values.

5.12.1 Detailed Description

Header file for the [Game](#) class.

Header file for the [HexTile](#) class.

5.12.2 Enumeration Type Documentation

5.12.2.1 TileResource

enum `TileResource`

An enumeration of the different tile resource values.

Enumerator

POOR	A poor resource value.
BELOW_AVERAGE	A below average resource value.
AVERAGE	An average resource value.
ABOVE_AVERAGE	An above average resource value.
GOOD	A good resource value.
N_TILE_RESOURCES	A simple hack to get the number of elements in TileResource.

```

54         {
55     POOR,
56     BELOW_AVERAGE,
57     AVERAGE,
58     ABOVE_AVERAGE,
59     GOOD,
60     N_TILE_RESOURCES
61 }; /* TileResource */

```

5.12.2.2 TileType

enum `TileType`

An enumeration of the different tile types.

Enumerator

NONE_TYPE	A dummy tile (for initialization).
FOREST	A forest tile.
LAKE	A lake tile.
MOUNTAINS	A mountains tile.
OCEAN	An ocean tile.
PLAINS	A plains tile.
N_TILE_TYPES	A simple hack to get the number of elements in TileType.

```

37         {
38     NONE_TYPE,
39     FOREST,
40     LAKE,
41     MOUNTAINS,
42     OCEAN,
43     PLAINS,
44     N_TILE_TYPES
45 }; /* TileType */

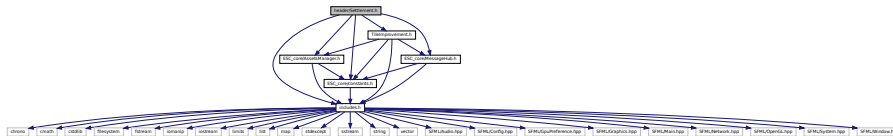
```

5.13 header/Settlement.h File Reference

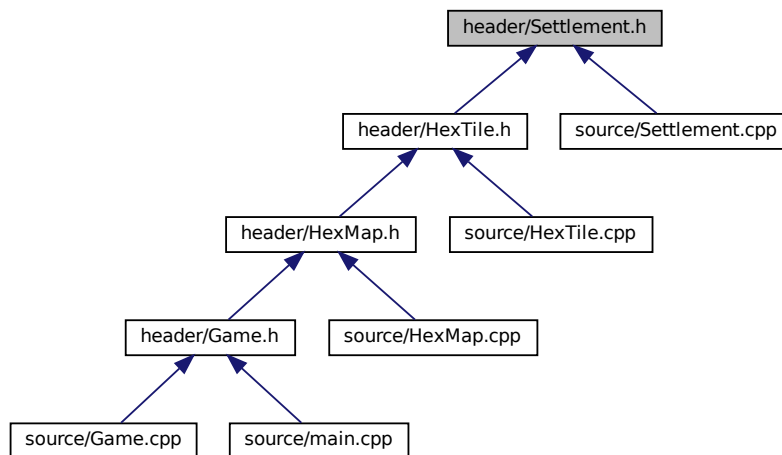
Header file for the [Settlement](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```

Include dependency graph for Settlement.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Settlement](#)
A settlement class (child class of [TileImprovement](#)).

5.13.1 Detailed Description

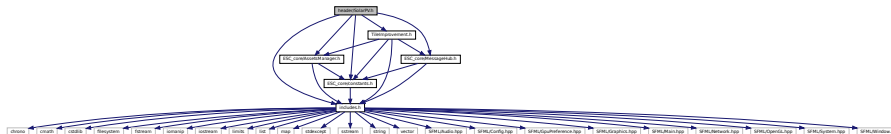
Header file for the [Settlement](#) class.

5.14 header/SolarPV.h File Reference

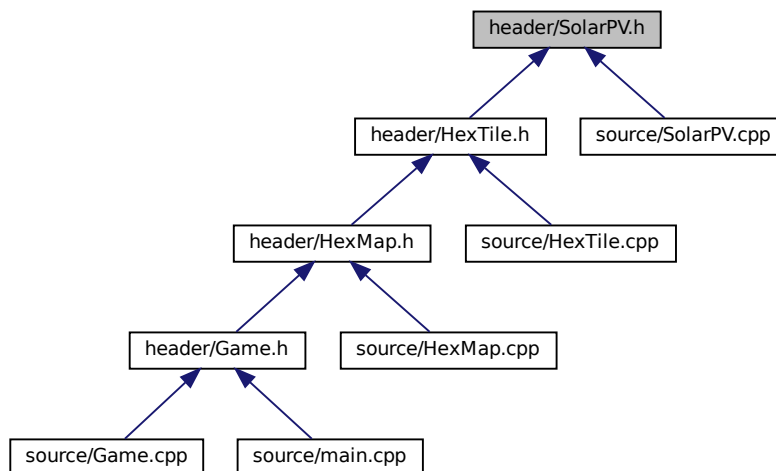
Header file for the [SolarPV](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```

Include dependency graph for SolarPV.h:



This graph shows which files directly or indirectly include this file:



Classes

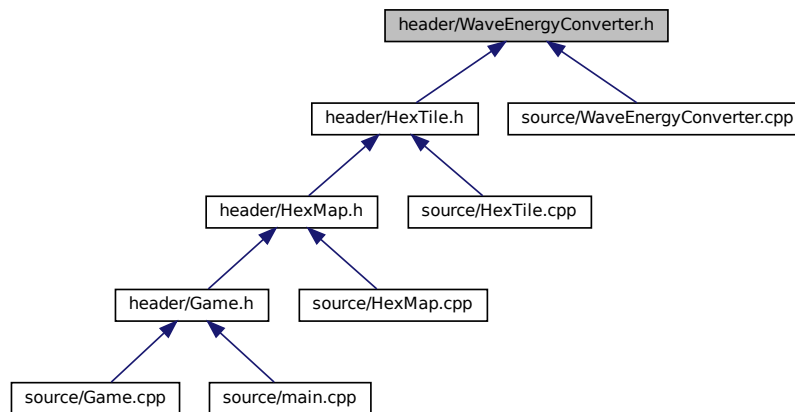
- class [SolarPV](#)

A settlement class (child class of [TileImprovement](#)).

5.14.1 Detailed Description

Header file for the [SolarPV](#) class.

This graph shows which files directly or indirectly include this file:



Classes

- class [WaveEnergyConverter](#)
A settlement class (child class of [TileImprovement](#)).

5.17.1 Detailed Description

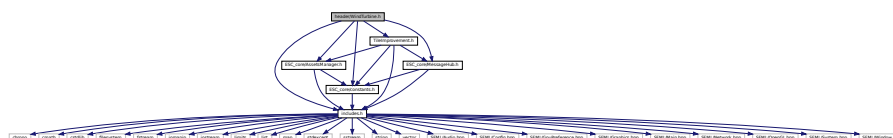
Header file for the [WaveEnergyConverter](#) class.

5.18 header/WindTurbine.h File Reference

Header file for the [WindTurbine](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```

Include dependency graph for `WindTurbine.h`:



Functions

- void `printGreen` (std::string input_str)
A function that sends green text to std::cout.
- void `printGold` (std::string input_str)
A function that sends gold text to std::cout.
- void `printRed` (std::string input_str)
A function that sends red text to std::cout.
- void `testFloatEquals` (double x, double y, std::string file, int line)
Tests for the equality of two floating point numbers x and y (to within FLOAT_TOLERANCE).
- void `testGreaterThan` (double x, double y, std::string file, int line)
Tests if $x > y$.
- void `testGreaterThanOrEqualTo` (double x, double y, std::string file, int line)
Tests if $x \geq y$.
- void `testLessThan` (double x, double y, std::string file, int line)
Tests if $x < y$.
- void `testLessThanOrEqualTo` (double x, double y, std::string file, int line)
Tests if $x \leq y$.
- void `testTruth` (bool statement, std::string file, int line)
Tests if the given statement is true.
- void `expectedErrorNotDetected` (std::string file, int line)
A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

5.24.1 Detailed Description

Implementation file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

5.24.2 Function Documentation

5.24.2.1 `expectedErrorNotDetected()`

```
void expectedErrorNotDetected (
    std::string file,
    int line )
```

A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

Parameters

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
```

```
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

5.24.2.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {
83     std::cout << "\x1B[33m" << input_str << "\033[0m";
84     return;
85 } /* printGold() */
```

5.24.2.3 printGreen()

```
void printGreen (
    std::string input_str )
```

A function that sends green text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
62 {
63     std::cout << "\x1B[32m" << input_str << "\033[0m";
64     return;
65 } /* printGreen() */
```

5.24.2.4 printRed()

```
void printRed (
    std::string input_str )
```

A function that sends red text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to <code>std::cout</code> .
------------------	---

```

102 {
103     std::cout << "\x1B[31m" << input_str << "\033[0m";
104     return;
105 } /* printRed() */

```

5.24.2.5 testFloatEquals()

```

void testFloatEquals (
    double x,
    double y,
    std::string file,
    int line )

```

Tests for the equality of two floating point numbers x and y (to within `FLOAT_TOLERANCE`).

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in " <code>__FILE__</code> ").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in " <code>__LINE__</code> ").

```

136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";
150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif
156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

5.24.2.6 testGreaterThan()

```

void testGreaterThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x > y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif
207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

5.24.2.7 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x \geq y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);

```

```

260     return;
261 } /* testGreaterThanOrEqualTo() */

```

5.24.2.8 testLessThan()

```

void testLessThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x < y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);
311     return;
312 } /* testLessThan() */

```

5.24.2.9 testLessThanOrEqualTo()

```

void testLessThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x \leq y$.

Parameters

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;
363 } /* testLessThanOrEqualTo() */

```

5.24.2.10 testTruth()

```

void testTruth (
    bool statement,
    std::string file,
    int line )

```

Tests if the given statement is true.

Parameters

<i>statement</i>	The statement whose truth is to be tested ("1 == 0", for example).
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\n";
400     error_str += "Given statement is not true";
401
402     #ifdef _WIN32
403         std::cout << error_str << std::endl;
404     #endif
405
406     throw std::runtime_error(error_str);
407     return;
408 } /* testTruth() */

```

5.25 source/Game.cpp File Reference

Implementation file for the [Game](#) class.

5.27.1 Detailed Description

Implementation file for the [HexTile](#) class.

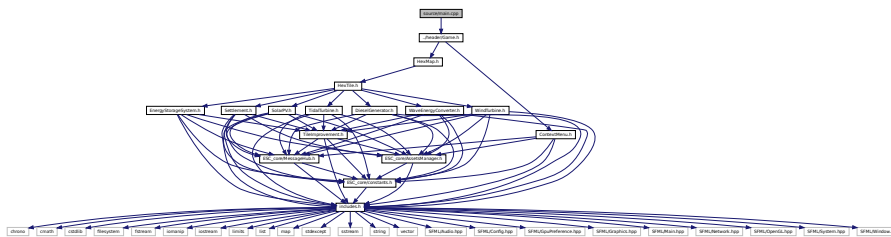
A class which defines a tile of a hex map.

5.28 source/main.cpp File Reference

Implementation file for [main\(\)](#) for Road To Zero.

```
#include "../header/Game.h"
```

Include dependency graph for main.cpp:



Functions

- void [loadAssets](#) ([AssetsManager](#) *assets_manager_ptr)
Helper function to load game assets.
- sf::RenderWindow * [constructRenderWindow](#) (void)
Helper function to construct render window.
- int [main](#) (int argc, char **argv)

5.28.1 Detailed Description

Implementation file for [main\(\)](#) for Road To Zero.

5.28.2 Function Documentation

5.28.2.1 constructRenderWindow()

```
sf::RenderWindow * constructRenderWindow (
    void )
```

Helper function to construct render window.

Returns

Pointer to the render window.

```
188 {
189     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
190         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
191         "Road To Zero"
192     );
193
194     return render_window_ptr;
195 } /* constructRenderWindow() */
```

5.28.2.2 loadAssets()

```
void loadAssets (
    AssetsManager * assets_manager_ptr )
```

Helper function to load game assets.

Parameters

<code>assets_manager_ptr</code>	Pointer to the assets manager.
---------------------------------	--------------------------------

```
32 {
33     // 1. load font assets
34     assets_manager_ptr->loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
35     assets_manager_ptr->loadFont("assets/fonts/Glass_TTY_VT220.ttf", "Glass_TTY_VT220");
36
37
38     // 2. load tile sheets
39     assets_manager_ptr->loadTexture(
40         "assets/tile_sheets/pine_tree_64x64_1.png",
41         "pine_tree_64x64_1"
42     );
43
44     assets_manager_ptr->loadTexture(
45         "assets/tile_sheets/wheat_64x64_1.png",
46         "wheat_64x64_1"
47     );
48
49     assets_manager_ptr->loadTexture(
50         "assets/tile_sheets/mountain_64x64_1.png",
51         "mountain_64x64_1"
52     );
53
54     assets_manager_ptr->loadTexture(
55         "assets/tile_sheets/water_waves_64x64_1.png",
56         "water_waves_64x64_1"
57     );
58
59     assets_manager_ptr->loadTexture(
60         "assets/tile_sheets/water_shimmer_64x64_1.png",
61         "water_shimmer_64x64_1"
62     );
63
64     assets_manager_ptr->loadTexture(
65         "assets/tile_sheets/brick_house_64x64_1.png",
66         "brick_house_64x64_1"
67     );
68
69     assets_manager_ptr->loadTexture(
70         "assets/tile_sheets/magnifying_glass_64x64_1.png",
71         "magnifying_glass_64x64_1"
72     );
73
74     assets_manager_ptr->loadTexture(
75         "assets/tile_sheets/exp2_0.png",
76         "tile clear explosion"
77     );
78
79     assets_manager_ptr->loadTexture(
80         "assets/tile_sheets/emissions_8x8_2.png",
81         "steam / smoke"
82     );
83
84     assets_manager_ptr->loadTexture(
85         "assets/tile_sheets/diesel_generator_64x64_2.png",
86         "diesel generator"
87     );
88
89     assets_manager_ptr->loadTexture(
90         "assets/tile_sheets/solar_PV_64x64_1.png",
91         "solar PV array"
92     );
93
94     assets_manager_ptr->loadTexture(
95         "assets/tile_sheets/wind_turbine_64x64_2.png",
96         "wind turbine"
97     );
98
99     assets_manager_ptr->loadTexture(
100         "assets/tile_sheets/energy_storage_system_64x64_1.png",
101         "energy storage system"
```

```

102     );
103
104     assets_manager_ptr->loadTexture(
105         "assets/tile_sheets/tidal_turbine_64x64_2.png",
106         "tidal turbine"
107     );
108
109     assets_manager_ptr->loadTexture(
110         "assets/tile_sheets/wave_energy_converter_64x64_2.png",
111         "wave energy converter"
112     );
113
114
115     // 3. load sounds
116     assets_manager_ptr->loadSound(
117         "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932.ogg",
118         "insufficient credits"
119     );
120
121     assets_manager_ptr->loadSound(
122         "assets/audio/samples/mixkit-sci-fi-click-900.ogg",
123         "resource assessment"
124     );
125
126     assets_manager_ptr->loadSound(
127         "assets/audio/samples/mixkit-interface-click-1126.ogg",
128         "console string print"
129     );
130
131     assets_manager_ptr->loadSound(
132         "assets/audio/samples/mixkit-video-game-retro-click-237.ogg",
133         "resource overlay toggle on"
134     );
135
136     assets_manager_ptr->loadSound(
137         "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED.ogg",
138         "resource overlay toggle off"
139     );
140
141     assets_manager_ptr->loadSound(
142         "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703.ogg",
143         "clear mountains tile"
144     );
145
146     assets_manager_ptr->loadSound(
147         "assets/audio/samples/mixkit-arcade-game-explosion-2759.ogg",
148         "clear non-mountains tile"
149     );
150
151     assets_manager_ptr->loadSound(
152         "assets/audio/samples/mixkit-electronic-retro-block-hit-2185.ogg",
153         "place improvement"
154     );
155
156     assets_manager_ptr->loadSound(
157         "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED.ogg",
158         "build menu open"
159     );
160
161     assets_manager_ptr->loadSound(
162         "assets/audio/samples/mixkit-video-game-lock-2851.ogg",
163         "build menu close"
164     );
165
166     assets_manager_ptr->loadSound(
167         "assets/audio/samples/mixkit-jump-into-the-water-1180.ogg",
168         "splash"
169     );
170
171     return;
172 } /* loadAssets() */

```

5.28.2.3 main()

```

int main (
    int argc,
    char ** argv )
204 {
205     // 1. load assets

```


Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL <https://www.sfml-dev.org/>. 193

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 192

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 39, 46, 91, 140, 146, 152, 158, 166, 172

Index

- __assembleHexMap
 - HexMap, [67](#)
- __assessNeighbours
 - HexMap, [68](#)
- __buildDieselGenerator
 - HexTile, [92](#)
- __buildDrawOrderVector
 - HexMap, [68](#)
- __buildEnergyStorage
 - HexTile, [92](#)
- __buildSettlement
 - HexTile, [93](#)
- __buildSolarPV
 - HexTile, [93](#)
- __buildTidalTurbine
 - HexTile, [94](#)
- __buildWaveEnergyConverter
 - HexTile, [95](#)
- __buildWindTurbine
 - HexTile, [95](#)
- __clearDecoration
 - HexTile, [96](#)
- __closeBuildMenu
 - HexTile, [96](#)
- __draw
 - Game, [52](#)
- __drawConsoleScreenFrame
 - ContextMenu, [22](#)
- __drawConsoleText
 - ContextMenu, [23](#)
- __drawFrameClockOverlay
 - Game, [52](#)
- __drawHUD
 - Game, [53](#)
- __drawVisualScreenFrame
 - ContextMenu, [24](#)
- __enforceOceanContinuity
 - HexMap, [69](#)
- __getMajorityTileType
 - HexMap, [70](#)
- __getNeighboursVector
 - HexMap, [71](#)
- __getNoise
 - HexMap, [71](#)
- __getSelectedTile
 - HexMap, [73](#)
- __getTileCoordsSubstring
 - HexTile, [97](#)
- __getTileImprovementSubstring
 - HexTile, [97](#)
- __getTileOptionsSubstring
 - HexTile, [97](#)
- __getTileResourceSubstring
 - HexTile, [99](#)
- __getTileTypeSubstring
 - HexTile, [100](#)
- __getValidMapIndexPositions
 - HexMap, [73](#)
- __handleKeyPressEvents
 - ContextMenu, [24](#)
 - DieselGenerator, [40](#)
 - EnergyStorageSystem, [47](#)
 - Game, [54](#)
 - HexMap, [74](#)
 - HexTile, [100](#)
 - Settlement, [141](#)
 - SolarPV, [147](#)
 - TidalTurbine, [153](#)
 - TileImprovement, [159](#)
 - WaveEnergyConverter, [167](#)
 - WindTurbine, [173](#)
- __handleMouseButtonEvents
 - ContextMenu, [25](#)
 - DieselGenerator, [40](#)
 - EnergyStorageSystem, [47](#)
 - Game, [55](#)
 - HexMap, [74](#)
 - HexTile, [104](#)
 - Settlement, [141](#)
 - SolarPV, [148](#)
 - TidalTurbine, [153](#)
 - TileImprovement, [159](#)
 - WaveEnergyConverter, [168](#)
 - WindTurbine, [173](#)
- __insufficientCreditsAlarm
 - Game, [55](#)
- __isClicked
 - HexTile, [105](#)
- __isLakeTouchingOcean
 - HexMap, [75](#)
- __layTiles
 - HexMap, [75](#)
- __loadSoundBuffer
 - AssetsManager, [9](#)
- __openBuildMenu
 - HexTile, [105](#)
- __procedurallyGenerateTileResources
 - HexMap, [78](#)

- __procedurallyGenerateTileTypes
 - HexMap, [78](#)
- __processEvent
 - Game, [57](#)
- __processMessage
 - Game, [57](#)
- __sendAssessNeighboursMessage
 - HexTile, [105](#)
- __sendCreditsSpentMessage
 - HexTile, [106](#)
- __sendGameStateMessage
 - Game, [58](#)
- __sendGameStateRequest
 - HexTile, [106](#)
- __sendInsufficientCreditsMessage
 - HexTile, [106](#)
- __sendNoTileSelectedMessage
 - HexMap, [79](#)
- __sendQuitGameMessage
 - ContextMenu, [25](#)
- __sendRestartGameMessage
 - ContextMenu, [25](#)
- __sendTileSelectedMessage
 - HexTile, [107](#)
- __sendTileStateMessage
 - HexTile, [107](#)
- __sendUpdateGamePhaseMessage
 - HexTile, [108](#)
- __setConsoleState
 - ContextMenu, [26](#)
- __setConsoleString
 - ContextMenu, [26](#)
- __setIsSelected
 - HexTile, [108](#)
- __setResourceText
 - HexTile, [108](#)
- __setUpBuildMenu
 - HexTile, [109](#)
- __setUpBuildOption
 - HexTile, [111](#)
- __setUpConsoleScreen
 - ContextMenu, [27](#)
- __setUpConsoleScreenFrame
 - ContextMenu, [27](#)
- __setUpDieselGeneratorBuildOption
 - HexTile, [112](#)
- __setUpEnergyStorageSystemBuildOption
 - HexTile, [112](#)
- __setUpGlassScreen
 - HexMap, [79](#)
- __setUpMagnifyingGlassSprite
 - HexTile, [112](#)
- __setUpMenuFrame
 - ContextMenu, [29](#)
- __setUpNodeSprite
 - HexTile, [113](#)
- __setUpResourceChipSprite
 - HexTile, [113](#)
- __setUpSelectOutlineSprite
 - HexTile, [113](#)
- __setUpSolarPVBuildOption
 - HexTile, [114](#)
- __setUpTidalTurbineBuildOption
 - HexTile, [115](#)
- __setUpTileExplosionReel
 - HexTile, [115](#)
- __setUpTileImprovementSpriteAnimated
 - DieselGenerator, [41](#)
 - TidalTurbine, [153](#)
 - WaveEnergyConverter, [168](#)
 - WindTurbine, [173](#)
- __setUpTileImprovementSpriteStatic
 - EnergyStorageSystem, [47](#)
 - Settlement, [141](#)
 - SolarPV, [148](#)
- __setUpTileSprite
 - HexTile, [115](#)
- __setUpVisualScreen
 - ContextMenu, [30](#)
- __setUpVisualScreenFrame
 - ContextMenu, [30](#)
- __setUpWaveEnergyConverterBuildOption
 - HexTile, [116](#)
- __setUpWindTurbineBuildOption
 - HexTile, [116](#)
- __smoothTileTypes
 - HexMap, [79](#)
- __toggleFrameClockOverlay
 - Game, [59](#)
- ~AssetsManager
 - AssetsManager, [8](#)
- ~ContextMenu
 - ContextMenu, [22](#)
- ~DieselGenerator
 - DieselGenerator, [40](#)
- ~EnergyStorageSystem
 - EnergyStorageSystem, [46](#)
- ~Game
 - Game, [52](#)
- ~HexMap
 - HexMap, [67](#)
- ~HexTile
 - HexTile, [92](#)
- ~MessageHub
 - MessageHub, [132](#)
- ~Settlement
 - Settlement, [140](#)
- ~SolarPV
 - SolarPV, [147](#)
- ~TidalTurbine
 - TidalTurbine, [152](#)
- ~TileImprovement
 - TileImprovement, [158](#)
- ~WaveEnergyConverter
 - WaveEnergyConverter, [167](#)
- ~WindTurbine

- WindTurbine, 172
- ABOVE_AVERAGE
 - HexTile.h, 204
- addChannel
 - MessageHub, 133
- assess
 - HexMap, 80
 - HexTile, 117
- assets_manager_ptr
 - ContextMenu, 33
 - Game, 60
 - HexMap, 83
 - HexTile, 123
 - TileImprovement, 162
- AssetsManager, 7
 - __loadSoundBuffer, 9
 - ~AssetsManager, 8
 - AssetsManager, 8
 - clear, 10
 - current_track, 18
 - font_map, 18
 - getCurrentTrackKey, 11
 - getFont, 11
 - getSound, 12
 - getSoundBuffer, 12
 - getTexture, 13
 - getTrackStatus, 13
 - loadFont, 14
 - loadSound, 14
 - loadTexture, 15
 - loadTrack, 16
 - nextTrack, 16
 - pauseTrack, 17
 - playTrack, 17
 - previousTrack, 17
 - sound_map, 18
 - soundbuffer_map, 18
 - stopTrack, 17
 - texture_map, 18
 - track_map, 19
- AVERAGE
 - HexTile.h, 204
- BELOW_AVERAGE
 - HexTile.h, 204
- bool_payload
 - Message, 130
- border_tiles_vec
 - HexMap, 83
- build_menu_backing
 - HexTile, 124
- build_menu_backing_text
 - HexTile, 124
- build_menu_open
 - HexTile, 124
- build_menu_options_text_vec
 - HexTile, 124
- build_menu_options_vec
 - HexTile, 124
- BUILD_SETTLEMENT
 - Game.h, 201
- BUILD_SETTLEMENT_COST
 - constants.h, 186
- channel
 - Message, 130
- clear
 - AssetsManager, 10
 - HexMap, 80
 - MessageHub, 133
- CLEAR_FOREST_COST
 - constants.h, 186
- CLEAR_MOUNTAINS_COST
 - constants.h, 186
- CLEAR_PLAINS_COST
 - constants.h, 186
- clearMessages
 - MessageHub, 134
- clock
 - Game, 60
- CO2E_KG_PER_LITRE_DIESEL
 - constants.h, 187
- console_screen
 - ContextMenu, 33
- console_screen_frame_bottom
 - ContextMenu, 33
- console_screen_frame_left
 - ContextMenu, 34
- console_screen_frame_right
 - ContextMenu, 34
- console_screen_frame_top
 - ContextMenu, 34
- console_state
 - ContextMenu, 34
- console_string
 - ContextMenu, 34
- console_string_changed
 - ContextMenu, 34
- console_substring_idx
 - ContextMenu, 35
- ConsoleState
 - ContextMenu.h, 178
- constants.h
 - BUILD_SETTLEMENT_COST, 186
 - CLEAR_FOREST_COST, 186
 - CLEAR_MOUNTAINS_COST, 186
 - CLEAR_PLAINS_COST, 186
 - CO2E_KG_PER_LITRE_DIESEL, 187
 - DIESEL_GENERATOR_BUILD_COST, 187
 - EMISSIONS_LIFETIME_LIMIT_TONNES, 187
 - ENERGY_STORAGE_SYSTEM_BUILD_COST, 187
 - FLOAT_TOLERANCE, 187
 - FOREST_GREEN, 183
 - FRAMES_PER_SECOND, 187
 - GAME_CHANNEL, 188
 - GAME_HEIGHT, 188

- GAME_STATE_CHANNEL, 188
- GAME_WIDTH, 188
- HEX_MAP_CHANNEL, 188
- LAKE_BLUE, 184
- MENU_FRAME_GREY, 184
- MONOCHROME_SCREEN_BACKGROUND, 184
- MONOCHROME_TEXT_AMBER, 184
- MONOCHROME_TEXT_GREEN, 184
- MONOCHROME_TEXT_RED, 185
- MOUNTAINS_GREY, 185
- NO_TILE_SELECTED_CHANNEL, 188
- OCEAN_BLUE, 185
- PLAINS_YELLOW, 185
- RESOURCE_ASSESSMENT_COST, 189
- RESOURCE_CHIP_GREY, 185
- SECONDS_PER_FRAME, 189
- SECONDS_PER_MONTH, 189
- SECONDS_PER_YEAR, 189
- SOLAR_PV_BUILD_COST, 189
- SOLAR_PV_WATER_BUILD_MULTIPLIER, 189
- STARTING_CREDITS, 190
- STARTING_POPULATION, 190
- TIDAL_TURBINE_BUILD_COST, 190
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES, ContextMenu.h
190
- TILE_SELECTED_CHANNEL, 190
- TILE_STATE_CHANNEL, 191
- TILE_TYPE_CUMULATIVE_PROBABILITIES, 191
- VISUAL_SCREEN_FRAME_GREY, 186
- WAVE_ENERGY_CONVERTER_BUILD_COST,
191
- WIND_TURBINE_BUILD_COST, 191
- WIND_TURBINE_WATER_BUILD_MULTIPLIER,
191
- constructRenderWindow
 - main.cpp, 221
- context_menu_ptr
 - Game, 61
- ContextMenu, 19
 - __drawConsoleScreenFrame, 22
 - __drawConsoleText, 23
 - __drawVisualScreenFrame, 24
 - __handleKeyPressEvents, 24
 - __handleMouseButtonEvents, 25
 - __sendQuitGameMessage, 25
 - __sendRestartGameMessage, 25
 - __setConsoleState, 26
 - __setConsoleString, 26
 - __setUpConsoleScreen, 27
 - __setUpConsoleScreenFrame, 27
 - __setUpMenuFrame, 29
 - __setUpVisualScreen, 30
 - __setUpVisualScreenFrame, 30
 - ~ContextMenu, 22
 - assets_manager_ptr, 33
 - console_screen, 33
 - console_screen_frame_bottom, 33
 - console_screen_frame_left, 34
 - console_screen_frame_right, 34
 - console_screen_frame_top, 34
 - console_state, 34
 - console_string, 34
 - console_string_changed, 34
 - console_substring_idx, 35
 - ContextMenu, 21
 - draw, 31
 - event_ptr, 35
 - frame, 35
 - game_menu_up, 35
 - menu_frame, 35
 - message_hub_ptr, 35
 - position_x, 36
 - position_y, 36
 - processEvent, 32
 - processMessage, 32
 - render_window_ptr, 36
 - visual_screen, 36
 - visual_screen_frame_bottom, 36
 - visual_screen_frame_left, 36
 - visual_screen_frame_right, 37
 - visual_screen_frame_top, 37
 - ConsoleState, 178
 - MENU, 178
 - N_CONSOLE_STATES, 178
 - NONE_STATE, 178
 - READY, 178
 - TILE, 178
- credits
 - Game, 61
 - HexTile, 124
 - TileImprovement, 162
- cumulative_emissions_tonnes
 - Game, 61
- current_track
 - AssetsManager, 18
- decorateTile
 - HexTile, 117
- decoration_cleared
 - HexTile, 125
- demand_MWh
 - Game, 61
- DIESEL_GENERATOR
 - TileImprovement.h, 209
- DIESEL_GENERATOR_BUILD_COST
 - constants.h, 187
- DieselGenerator, 37
 - __handleKeyPressEvents, 40
 - __handleMouseButtonEvents, 40
 - __setUpTileImprovementSpriteAnimated, 41
 - ~DieselGenerator, 40
 - DieselGenerator, 39
 - draw, 41
 - processEvent, 42
 - processMessage, 42
 - skip_smoke_processing, 43

- smoke_da, 43
- smoke_dx, 43
- smoke_dy, 43
- smoke_prob, 43
- smoke_sprite_list, 44
- double_payload
 - Message, 130
- draw
 - ContextMenu, 31
 - DieselGenerator, 41
 - EnergyStorageSystem, 48
 - HexMap, 81
 - HexTile, 118
 - Settlement, 142
 - SolarPV, 149
 - TidalTurbine, 154
 - TileImprovement, 160
 - WaveEnergyConverter, 169
 - WindTurbine, 174
- draw_explosion
 - HexTile, 125
- EMISSIONS_LIFETIME_LIMIT_TONNES
 - constants.h, 187
- ENERGY_STORAGE_SYSTEM
 - TileImprovement.h, 209
- ENERGY_STORAGE_SYSTEM_BUILD_COST
 - constants.h, 187
- EnergyStorageSystem, 44
 - __handleKeyPressEvents, 47
 - __handleMouseButtonEvents, 47
 - __setUpTileImprovementSpriteStatic, 47
 - ~EnergyStorageSystem, 46
 - draw, 48
 - EnergyStorageSystem, 46
 - processEvent, 48
 - processMessage, 48
- event
 - Game, 61
- event_ptr
 - ContextMenu, 35
 - HexMap, 84
 - HexTile, 125
 - TileImprovement, 162
- expectedErrorNotDetected
 - testing_utils.cpp, 214
 - testing_utils.h, 195
- explosion_frame
 - HexTile, 125
- explosion_sprite_reel
 - HexTile, 125
- FLOAT_TOLERANCE
 - constants.h, 187
- font_map
 - AssetsManager, 18
- FOREST
 - HexTile.h, 204
- FOREST_GREEN
 - constants.h, 183
- frame
 - ContextMenu, 35
 - Game, 61
 - HexMap, 84
 - HexTile, 125
 - TileImprovement, 162
- FRAMES_PER_SECOND
 - constants.h, 187
- Game, 49
 - __draw, 52
 - __drawFrameClockOverlay, 52
 - __drawHUD, 53
 - __handleKeyPressEvents, 54
 - __handleMouseButtonEvents, 55
 - __insufficientCreditsAlarm, 55
 - __processEvent, 57
 - __processMessage, 57
 - __sendGameStateMessage, 58
 - __toggleFrameClockOverlay, 59
 - ~Game, 52
 - assets_manager_ptr, 60
 - clock, 60
 - context_menu_ptr, 61
 - credits, 61
 - cumulative_emissions_tonnes, 61
 - demand_MWh, 61
 - event, 61
 - frame, 61
 - Game, 51
 - game_loop_broken, 62
 - game_phase, 62
 - hex_map_ptr, 62
 - message_hub, 62
 - month, 62
 - population, 62
 - quit_game, 63
 - render_window_ptr, 63
 - run, 59
 - show_frame_clock_overlay, 63
 - time_since_start_s, 63
 - turn, 63
 - year, 63
- Game.h
 - BUILD_SETTLEMENT, 201
 - GamePhase, 201
 - LOSS_CREDITS, 201
 - LOSS_DEMAND, 201
 - LOSS_EMISSIONS, 201
 - N_GAME_PHASES, 201
 - SYSTEM_MANAGEMENT, 201
 - VICTORY, 201
- GAME_CHANNEL
 - constants.h, 188
- GAME_HEIGHT
 - constants.h, 188
- game_loop_broken
 - Game, 62

game_menu_up
 ContextMenu, 35
 game_phase
 Game, 62
 HexTile, 126
 TileImprovement, 163
 GAME_STATE_CHANNEL
 constants.h, 188
 GAME_WIDTH
 constants.h, 188
 GamePhase
 Game.h, 201
 getCurrentTrackKey
 AssetsManager, 11
 getFont
 AssetsManager, 11
 getSound
 AssetsManager, 12
 getSoundBuffer
 AssetsManager, 12
 getTexture
 AssetsManager, 13
 getTrackStatus
 AssetsManager, 13
 glass_screen
 HexMap, 84
 GOOD
 HexTile.h, 204

 has_improvement
 HexTile, 126
 hasTraffic
 MessageHub, 134
 header/ContextMenu.h, 177
 header/DieselGenerator.h, 178
 header/EnergyStorageSystem.h, 179
 header/ESC_core/AssetsManager.h, 180
 header/ESC_core/constants.h, 181
 header/ESC_core/doxygen_cite.h, 192
 header/ESC_core/includes.h, 192
 header/ESC_core/MessageHub.h, 193
 header/ESC_core/testing_utils.h, 194
 header/Game.h, 200
 header/HexMap.h, 201
 header/HexTile.h, 202
 header/Settlement.h, 205
 header/SolarPV.h, 206
 header/TidalTurbine.h, 207
 header/TileImprovement.h, 208
 header/WaveEnergyConverter.h, 209
 header/WindTurbine.h, 210
 hex_draw_order_vec
 HexMap, 84
 hex_map
 HexMap, 84
 HEX_MAP_CHANNEL
 constants.h, 188
 hex_map_ptr
 Game, 62

 HexMap, 64
 __assembleHexMap, 67
 __assessNeighbours, 68
 __buildDrawOrderVector, 68
 __enforceOceanContinuity, 69
 __getMajorityTileType, 70
 __getNeighboursVector, 71
 __getNoise, 71
 __getSelectedTile, 73
 __getValidMapIndexPositions, 73
 __handleKeyPressEvents, 74
 __handleMouseButtonEvents, 74
 __isLakeTouchingOcean, 75
 __layTiles, 75
 __procedurallyGenerateTileResources, 78
 __procedurallyGenerateTileTypes, 78
 __sendNoTileSelectedMessage, 79
 __setUpGlassScreen, 79
 __smoothTileTypes, 79
 ~HexMap, 67
 assess, 80
 assets_manager_ptr, 83
 border_tiles_vec, 83
 clear, 80
 draw, 81
 event_ptr, 84
 frame, 84
 glass_screen, 84
 hex_draw_order_vec, 84
 hex_map, 84
 HexMap, 66
 message_hub_ptr, 84
 n_layers, 85
 n_tiles, 85
 position_x, 85
 position_y, 85
 processEvent, 81
 processMessage, 82
 render_window_ptr, 85
 reroll, 82
 show_resource, 85
 tile_position_x_vec, 86
 tile_position_y_vec, 86
 tile_selected, 86
 toggleResourceOverlay, 83
 HexTile, 86
 __buildDieselGenerator, 92
 __buildEnergyStorage, 92
 __buildSettlement, 93
 __buildSolarPV, 93
 __buildTidalTurbine, 94
 __buildWaveEnergyConverter, 95
 __buildWindTurbine, 95
 __clearDecoration, 96
 __closeBuildMenu, 96
 __getTileCoordsSubstring, 97
 __getTileImprovementSubstring, 97
 __getTileOptionsSubstring, 97

- __getTileResourceSubstring, 99
- __getTileTypeSubstring, 100
- __handleKeyPressEvents, 100
- __handleMouseButtonEvents, 104
- __isClicked, 105
- __openBuildMenu, 105
- __sendAssessNeighboursMessage, 105
- __sendCreditsSpentMessage, 106
- __sendGameStateRequest, 106
- __sendInsufficientCreditsMessage, 106
- __sendTileSelectedMessage, 107
- __sendTileStateMessage, 107
- __sendUpdateGamePhaseMessage, 108
- __setIsSelected, 108
- __setResourceText, 108
- __setUpBuildMenu, 109
- __setUpBuildOption, 111
- __setUpDieselGeneratorBuildOption, 112
- __setUpEnergyStorageSystemBuildOption, 112
- __setUpMagnifyingGlassSprite, 112
- __setUpNodeSprite, 113
- __setUpResourceChipSprite, 113
- __setUpSelectOutlineSprite, 113
- __setUpSolarPVBuildOption, 114
- __setUpTidalTurbineBuildOption, 115
- __setUpTileExplosionReel, 115
- __setUpTileSprite, 115
- __setUpWaveEnergyConverterBuildOption, 116
- __setUpWindTurbineBuildOption, 116
- ~HexTile, 92
- assess, 117
- assets_manager_ptr, 123
- build_menu_backing, 124
- build_menu_backing_text, 124
- build_menu_open, 124
- build_menu_options_text_vec, 124
- build_menu_options_vec, 124
- credits, 124
- decorateTile, 117
- decoration_cleared, 125
- draw, 118
- draw_explosion, 125
- event_ptr, 125
- explosion_frame, 125
- explosion_sprite_reel, 125
- frame, 125
- game_phase, 126
- has_improvement, 126
- HexTile, 90
- is_selected, 126
- magnifying_glass_sprite, 126
- major_radius, 126
- message_hub_ptr, 126
- minor_radius, 127
- node_sprite, 127
- position_x, 127
- position_y, 127
- processEvent, 120
- processMessage, 120
- render_window_ptr, 127
- resource_assessed, 127
- resource_assessment, 128
- resource_chip_sprite, 128
- resource_text, 128
- select_outline_sprite, 128
- setTileResource, 121
- setTileType, 122
- show_node, 128
- show_resource, 128
- tile_decoration_sprite, 129
- tile_improvement_ptr, 129
- tile_resource, 129
- tile_sprite, 129
- tile_type, 129
- toggleResourceOverlay, 123
- HexTile.h
 - ABOVE_AVERAGE, 204
 - AVERAGE, 204
 - BELOW_AVERAGE, 204
 - FOREST, 204
 - GOOD, 204
 - LAKE, 204
 - MOUNTAINS, 204
 - N_TILE_RESOURCES, 204
 - N_TILE_TYPES, 204
 - NONE_TYPE, 204
 - OCEAN, 204
 - PLAINS, 204
 - POOR, 204
 - TileResource, 204
 - TileType, 204
- int_payload
 - Message, 131
- is_running
 - TileImprovement, 163
- is_selected
 - HexTile, 126
 - TileImprovement, 163
- isEmpty
 - MessageHub, 134
- just_built
 - TileImprovement, 163
- LAKE
 - HexTile.h, 204
- LAKE_BLUE
 - constants.h, 184
- loadAssets
 - main.cpp, 221
- loadFont
 - AssetsManager, 14
- loadSound
 - AssetsManager, 14
- loadTexture
 - AssetsManager, 15

- loadTrack
 - AssetsManager, 16
- LOSS_CREDITS
 - Game.h, 201
- LOSS_DEMAND
 - Game.h, 201
- LOSS_EMISSIONS
 - Game.h, 201
- magnifying_glass_sprite
 - HexTile, 126
- main
 - main.cpp, 223
- main.cpp
 - constructRenderWindow, 221
 - loadAssets, 221
 - main, 223
- major_radius
 - HexTile, 126
- MENU
 - ContextMenu.h, 178
- menu_frame
 - ContextMenu, 35
- MENU_FRAME_GREY
 - constants.h, 184
- Message, 130
 - bool_payload, 130
 - channel, 130
 - double_payload, 130
 - int_payload, 131
 - string_payload, 131
 - subject, 131
- message_hub
 - Game, 62
- message_hub_ptr
 - ContextMenu, 35
 - HexMap, 84
 - HexTile, 126
 - TileImprovement, 163
- message_map
 - MessageHub, 137
- MessageHub, 131
 - ~MessageHub, 132
 - addChannel, 133
 - clear, 133
 - clearMessages, 134
 - hasTraffic, 134
 - isEmpty, 134
 - message_map, 137
 - MessageHub, 132
 - popMessage, 135
 - receiveMessage, 135
 - removeChannel, 136
 - sendMessage, 137
- minor_radius
 - HexTile, 127
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, 184
- MONOCHROME_TEXT_AMBER
 - constants.h, 184
- MONOCHROME_TEXT_GREEN
 - constants.h, 184
- MONOCHROME_TEXT_RED
 - constants.h, 185
- month
 - Game, 62
- MOUNTAINS
 - HexTile.h, 204
- MOUNTAINS_GREY
 - constants.h, 185
- N_CONSOLE_STATES
 - ContextMenu.h, 178
- N_GAME_PHASES
 - Game.h, 201
- n_layers
 - HexMap, 85
- N_TILE_IMPROVEMENT_TYPES
 - TileImprovement.h, 209
- N_TILE_RESOURCES
 - HexTile.h, 204
- N_TILE_TYPES
 - HexTile.h, 204
- n_tiles
 - HexMap, 85
- nextTrack
 - AssetsManager, 16
- NO_TILE_SELECTED_CHANNEL
 - constants.h, 188
- node_sprite
 - HexTile, 127
- NONE_STATE
 - ContextMenu.h, 178
- NONE_TYPE
 - HexTile.h, 204
- OCEAN
 - HexTile.h, 204
- OCEAN_BLUE
 - constants.h, 185
- pauseTrack
 - AssetsManager, 17
- PLAINS
 - HexTile.h, 204
- PLAINS_YELLOW
 - constants.h, 185
- playTrack
 - AssetsManager, 17
- POOR
 - HexTile.h, 204
- popMessage
 - MessageHub, 135
- population
 - Game, 62
- position_x
 - ContextMenu, 36
 - HexMap, 85

- HexTile, 127
- TileImprovement, 163
- position_y
 - ContextMenu, 36
 - HexMap, 85
 - HexTile, 127
 - TileImprovement, 164
- previousTrack
 - AssetsManager, 17
- printGold
 - testing_utils.cpp, 215
 - testing_utils.h, 195
- printGreen
 - testing_utils.cpp, 215
 - testing_utils.h, 195
- printRed
 - testing_utils.cpp, 215
 - testing_utils.h, 196
- processEvent
 - ContextMenu, 32
 - DieselGenerator, 42
 - EnergyStorageSystem, 48
 - HexMap, 81
 - HexTile, 120
 - Settlement, 143
 - SolarPV, 149
 - TidalTurbine, 154
 - TileImprovement, 161
 - WaveEnergyConverter, 169
 - WindTurbine, 174
- processMessage
 - ContextMenu, 32
 - DieselGenerator, 42
 - EnergyStorageSystem, 48
 - HexMap, 82
 - HexTile, 120
 - Settlement, 143
 - SolarPV, 149
 - TidalTurbine, 155
 - TileImprovement, 162
 - WaveEnergyConverter, 170
 - WindTurbine, 175
- quit_game
 - Game, 63
- READY
 - ContextMenu.h, 178
- receiveMessage
 - MessageHub, 135
- removeChannel
 - MessageHub, 136
- render_window_ptr
 - ContextMenu, 36
 - Game, 63
 - HexMap, 85
 - HexTile, 127
 - TileImprovement, 164
- reroll
 - HexMap, 82
 - resource_assessed
 - HexTile, 127
 - resource_assessment
 - HexTile, 128
 - RESOURCE_ASSESSMENT_COST
 - constants.h, 189
 - RESOURCE_CHIP_GREY
 - constants.h, 185
 - resource_chip_sprite
 - HexTile, 128
 - resource_text
 - HexTile, 128
 - run
 - Game, 59
 - SECONDS_PER_FRAME
 - constants.h, 189
 - SECONDS_PER_MONTH
 - constants.h, 189
 - SECONDS_PER_YEAR
 - constants.h, 189
 - select_outline_sprite
 - HexTile, 128
 - sendMessage
 - MessageHub, 137
 - setTileResource
 - HexTile, 121
 - setTileType
 - HexTile, 122
 - SETTLEMENT
 - TileImprovement.h, 209
 - Settlement, 138
 - __handleKeyPressEvents, 141
 - __handleMouseButtonEvents, 141
 - __setUpTileImprovementSpriteStatic, 141
 - ~Settlement, 140
 - draw, 142
 - processEvent, 143
 - processMessage, 143
 - Settlement, 139
 - skip_smoke_processing, 144
 - smoke_da, 144
 - smoke_dx, 144
 - smoke_dy, 144
 - smoke_prob, 144
 - smoke_sprite_list, 144
 - show_frame_clock_overlay
 - Game, 63
 - show_node
 - HexTile, 128
 - show_resource
 - HexMap, 85
 - HexTile, 128
 - skip_smoke_processing
 - DieselGenerator, 43
 - Settlement, 144
 - smoke_da
 - DieselGenerator, 43

- Settlement, 144
- smoke_dx
 - DieselGenerator, 43
 - Settlement, 144
- smoke_dy
 - DieselGenerator, 43
 - Settlement, 144
- smoke_prob
 - DieselGenerator, 43
 - Settlement, 144
- smoke_sprite_list
 - DieselGenerator, 44
 - Settlement, 144
- SOLAR_PV
 - TileImprovement.h, 209
- SOLAR_PV_BUILD_COST
 - constants.h, 189
- SOLAR_PV_WATER_BUILD_MULTIPLIER
 - constants.h, 189
- SolarPV, 145
 - __handleKeyPressEvents, 147
 - __handleMouseButtonEvents, 148
 - __setUpTileImprovementSpriteStatic, 148
 - ~SolarPV, 147
 - draw, 149
 - processEvent, 149
 - processMessage, 149
 - SolarPV, 146
- sound_map
 - AssetsManager, 18
- soundbuffer_map
 - AssetsManager, 18
- source/ContextMenu.cpp, 211
- source/DieselGenerator.cpp, 212
- source/EnergyStorageSystem.cpp, 212
- source/ESC_core/AssetsManager.cpp, 212
- source/ESC_core/MessageHub.cpp, 213
- source/ESC_core/testing_utils.cpp, 213
- source/Game.cpp, 219
- source/HexMap.cpp, 220
- source/HexTile.cpp, 220
- source/main.cpp, 221
- source/Settlement.cpp, 224
- source/SolarPV.cpp, 224
- source/TidalTurbine.cpp, 225
- source/TileImprovement.cpp, 225
- source/WaveEnergyConverter.cpp, 226
- source/WindTurbine.cpp, 226
- STARTING_CREDITS
 - constants.h, 190
- STARTING_POPULATION
 - constants.h, 190
- stopTrack
 - AssetsManager, 17
- string_payload
 - Message, 131
- subject
 - Message, 131
- SYSTEM_MANAGEMENT
 - Game.h, 201
- testFloatEquals
 - testing_utils.cpp, 216
 - testing_utils.h, 196
- testGreaterThan
 - testing_utils.cpp, 216
 - testing_utils.h, 197
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 217
 - testing_utils.h, 197
- testing_utils.cpp
 - expectedErrorNotDetected, 214
 - printGold, 215
 - printGreen, 215
 - printRed, 215
 - testFloatEquals, 216
 - testGreaterThan, 216
 - testGreaterThanOrEqualTo, 217
 - testLessThan, 218
 - testLessThanOrEqualTo, 218
 - testTruth, 219
- testing_utils.h
 - expectedErrorNotDetected, 195
 - printGold, 195
 - printGreen, 195
 - printRed, 196
 - testFloatEquals, 196
 - testGreaterThan, 197
 - testGreaterThanOrEqualTo, 197
 - testLessThan, 198
 - testLessThanOrEqualTo, 199
 - testTruth, 199
- testLessThan
 - testing_utils.cpp, 218
 - testing_utils.h, 198
- testLessThanOrEqualTo
 - testing_utils.cpp, 218
 - testing_utils.h, 199
- testTruth
 - testing_utils.cpp, 219
 - testing_utils.h, 199
- texture_map
 - AssetsManager, 18
- TIDAL_TURBINE
 - TileImprovement.h, 209
- TIDAL_TURBINE_BUILD_COST
 - constants.h, 190
- TidalTurbine, 150
 - __handleKeyPressEvents, 153
 - __handleMouseButtonEvents, 153
 - __setUpTileImprovementSpriteAnimated, 153
 - ~TidalTurbine, 152
 - draw, 154
 - processEvent, 154
 - processMessage, 155
 - TidalTurbine, 152
- TILE

- ContextMenu.h, 178
- tile_decoration_sprite
 - HexTile, 129
- tile_improvement_ptr
 - HexTile, 129
- tile_improvement_sprite_animated
 - TileImprovement, 164
- tile_improvement_sprite_static
 - TileImprovement, 164
- tile_improvement_string
 - TileImprovement, 164
- tile_improvement_type
 - TileImprovement, 164
- tile_position_x_vec
 - HexMap, 86
- tile_position_y_vec
 - HexMap, 86
- tile_resource
 - HexTile, 129
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES
 - constants.h, 190
- tile_selected
 - HexMap, 86
- TILE_SELECTED_CHANNEL
 - constants.h, 190
- tile_sprite
 - HexTile, 129
- TILE_STATE_CHANNEL
 - constants.h, 191
- tile_type
 - HexTile, 129
- TILE_TYPE_CUMULATIVE_PROBABILITIES
 - constants.h, 191
- TileImprovement, 155
 - __handleKeyPressEvents, 159
 - __handleMouseButtonEvents, 159
 - ~TileImprovement, 158
 - assets_manager_ptr, 162
 - credits, 162
 - draw, 160
 - event_ptr, 162
 - frame, 162
 - game_phase, 163
 - is_running, 163
 - is_selected, 163
 - just_built, 163
 - message_hub_ptr, 163
 - position_x, 163
 - position_y, 164
 - processEvent, 161
 - processMessage, 162
 - render_window_ptr, 164
 - tile_improvement_sprite_animated, 164
 - tile_improvement_sprite_static, 164
 - tile_improvement_string, 164
 - tile_improvement_type, 164
 - TileImprovement, 158
- TileImprovement.h
 - DIESEL_GENERATOR, 209
 - ENERGY_STORAGE_SYSTEM, 209
 - N_TILE_IMPROVEMENT_TYPES, 209
 - SETTLEMENT, 209
 - SOLAR_PV, 209
 - TIDAL_TURBINE, 209
 - TileImprovementType, 208
 - WAVE_ENERGY_CONVERTER, 209
 - WIND_TURBINE, 209
- TileImprovementType
 - TileImprovement.h, 208
- TileResource
 - HexTile.h, 204
- TileType
 - HexTile.h, 204
- time_since_start_s
 - Game, 63
- toggleResourceOverlay
 - HexMap, 83
 - HexTile, 123
- track_map
 - AssetsManager, 19
- turn
 - Game, 63
- VICTORY
 - Game.h, 201
- visual_screen
 - ContextMenu, 36
- visual_screen_frame_bottom
 - ContextMenu, 36
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, 186
- visual_screen_frame_left
 - ContextMenu, 36
- visual_screen_frame_right
 - ContextMenu, 37
- visual_screen_frame_top
 - ContextMenu, 37
- WAVE_ENERGY_CONVERTER
 - TileImprovement.h, 209
- WAVE_ENERGY_CONVERTER_BUILD_COST
 - constants.h, 191
- WaveEnergyConverter, 165
 - __handleKeyPressEvents, 167
 - __handleMouseButtonEvents, 168
 - __setUpTileImprovementSpriteAnimated, 168
 - ~WaveEnergyConverter, 167
 - draw, 169
 - processEvent, 169
 - processMessage, 170
 - WaveEnergyConverter, 166
- WIND_TURBINE
 - TileImprovement.h, 209
- WIND_TURBINE_BUILD_COST
 - constants.h, 191
- WIND_TURBINE_WATER_BUILD_MULTIPLIER
 - constants.h, 191

WindTurbine, [170](#)
 __handleKeyPressEvents, [173](#)
 __handleMouseButtonEvents, [173](#)
 __setUpTileImprovementSpriteAnimated, [173](#)
 ~WindTurbine, [172](#)
 draw, [174](#)
 processEvent, [174](#)
 processMessage, [175](#)
 WindTurbine, [172](#)

year
 Game, [63](#)