

Road To Zero - The Microgrid Management Game

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 getTileOptionsSubstring()	42
4.3.3.6 processEvent()	43
4.3.3.7 processMessage()	43
4.3.4 Member Data Documentation	44
4.3.4.1 capacity_kW	44
4.3.4.2 max_production_MWh	44
4.3.4.3 production_MWh	44
4.3.4.4 smoke_da	44
4.3.4.5 smoke_dx	44
4.3.4.6 smoke_dy	45
4.3.4.7 smoke_prob	45
4.3.4.8 smoke_sprite_list	45
4.4 EnergyStorageSystem Class Reference	45
4.4.1 Detailed Description	46
4.4.2 Constructor & Destructor Documentation	47
4.4.2.1 EnergyStorageSystem()	47
4.4.2.2 ~EnergyStorageSystem()	47
4.4.3 Member Function Documentation	48
4.4.3.1 __handleKeyPressEvents()	48
4.4.3.2 __handleMouseButtonEvents()	48
4.4.3.3 __setUpTileImprovementSpriteStatic()	49
4.4.3.4 draw()	49
4.4.3.5 getTileOptionsSubstring()	49
4.4.3.6 processEvent()	50
4.4.3.7 processMessage()	50
4.5 Game Class Reference	51
4.5.1 Detailed Description	53
4.5.2 Constructor & Destructor Documentation	53
4.5.2.1 Game()	53
4.5.2.2 ~Game()	54

4.5.3 Member Function Documentation	54
4.5.3.1 __draw()	54
4.5.3.2 __drawFrameClockOverlay()	54
4.5.3.3 __drawHUD()	55
4.5.3.4 __handleKeyPressEvents()	56
4.5.3.5 __handleMouseButtonEvents()	57
4.5.3.6 __insufficientCreditsAlarm()	57
4.5.3.7 __processEvent()	58
4.5.3.8 __processMessage()	59
4.5.3.9 __sendGameStateMessage()	60
4.5.3.10 __toggleFrameClockOverlay()	61
4.5.3.11 run()	61
4.5.4 Member Data Documentation	62
4.5.4.1 assets_manager_ptr	62
4.5.4.2 clock	62
4.5.4.3 context_menu_ptr	62
4.5.4.4 credits	62
4.5.4.5 cumulative_emissions_tonnes	63
4.5.4.6 demand_MWh	63
4.5.4.7 event	63
4.5.4.8 frame	63
4.5.4.9 game_loop_broken	63
4.5.4.10 game_phase	63
4.5.4.11 hex_map_ptr	64
4.5.4.12 message_hub	64
4.5.4.13 month	64
4.5.4.14 population	64
4.5.4.15 quit_game	64
4.5.4.16 render_window_ptr	64
4.5.4.17 show_frame_clock_overlay	65
4.5.4.18 time_since_start_s	65
4.5.4.19 turn	65
4.5.4.20 year	65
4.6 HexMap Class Reference	65
4.6.1 Detailed Description	68
4.6.2 Constructor & Destructor Documentation	68
4.6.2.1 HexMap()	68
4.6.2.2 ~HexMap()	69
4.6.3 Member Function Documentation	69
4.6.3.1 __assembleHexMap()	69
4.6.3.2 __assessNeighbours()	69
4.6.3.3 __buildDrawOrderVector()	70

4.6.3.4	__enforceOceanContinuity()	71
4.6.3.5	__getMajorityTileType()	71
4.6.3.6	__getNeighboursVector()	72
4.6.3.7	__getNoise()	73
4.6.3.8	__getSelectedTile()	74
4.6.3.9	__getValidMapIndexPositions()	75
4.6.3.10	__handleKeyPressEvents()	76
4.6.3.11	__handleMouseButtonEvents()	76
4.6.3.12	__isLakeTouchingOcean()	77
4.6.3.13	__layTiles()	77
4.6.3.14	__procedurallyGenerateTileResources()	79
4.6.3.15	__procedurallyGenerateTileTypes()	80
4.6.3.16	__sendNoTileSelectedMessage()	81
4.6.3.17	__setUpGlassScreen()	81
4.6.3.18	__smoothTileTypes()	81
4.6.3.19	assess()	82
4.6.3.20	clear()	82
4.6.3.21	draw()	82
4.6.3.22	processEvent()	83
4.6.3.23	processMessage()	84
4.6.3.24	reroll()	84
4.6.3.25	toggleResourceOverlay()	84
4.6.4	Member Data Documentation	85
4.6.4.1	assets_manager_ptr	85
4.6.4.2	border_tiles_vec	85
4.6.4.3	event_ptr	85
4.6.4.4	frame	85
4.6.4.5	glass_screen	86
4.6.4.6	hex_draw_order_vec	86
4.6.4.7	hex_map	86
4.6.4.8	message_hub_ptr	86
4.6.4.9	n_layers	86
4.6.4.10	n_tiles	86
4.6.4.11	position_x	87
4.6.4.12	position_y	87
4.6.4.13	render_window_ptr	87
4.6.4.14	show_resource	87
4.6.4.15	tile_position_x_vec	87
4.6.4.16	tile_position_y_vec	87
4.6.4.17	tile_selected	88
4.7	HexTile Class Reference	88
4.7.1	Detailed Description	92

4.7.2 Constructor & Destructor Documentation	92
4.7.2.1 HexTile()	92
4.7.2.2 ~HexTile()	93
4.7.3 Member Function Documentation	94
4.7.3.1 __buildDieselGenerator()	94
4.7.3.2 __buildEnergyStorage()	94
4.7.3.3 __buildSettlement()	95
4.7.3.4 __buildSolarPV()	95
4.7.3.5 __buildTidalTurbine()	96
4.7.3.6 __buildWaveEnergyConverter()	97
4.7.3.7 __buildWindTurbine()	97
4.7.3.8 __clearDecoration()	98
4.7.3.9 __closeBuildMenu()	98
4.7.3.10 __getTileCoordsSubstring()	99
4.7.3.11 __getTileImprovementSubstring()	99
4.7.3.12 __getTileOptionsSubstring()	99
4.7.3.13 __getTileResourceSubstring()	101
4.7.3.14 __getTileTypeSubstring()	102
4.7.3.15 __handleKeyPressEvents()	102
4.7.3.16 __handleMouseButtonEvents()	106
4.7.3.17 __isClicked()	107
4.7.3.18 __openBuildMenu()	107
4.7.3.19 __scrapImprovement()	108
4.7.3.20 __sendAssessNeighboursMessage()	108
4.7.3.21 __sendCreditsSpentMessage()	108
4.7.3.22 __sendGameStateRequest()	109
4.7.3.23 __sendInsufficientCreditsMessage()	109
4.7.3.24 __sendTileSelectedMessage()	109
4.7.3.25 __sendTileStateMessage()	110
4.7.3.26 __sendUpdateGamePhaseMessage()	110
4.7.3.27 __setIsSelected()	110
4.7.3.28 __setResourceText()	111
4.7.3.29 __setUpBuildMenu()	112
4.7.3.30 __setUpBuildOption()	113
4.7.3.31 __setUpDieselGeneratorBuildOption()	114
4.7.3.32 __setUpEnergyStorageSystemBuildOption()	115
4.7.3.33 __setUpMagnifyingGlassSprite()	115
4.7.3.34 __setUpNodeSprite()	115
4.7.3.35 __setUpResourceChipSprite()	116
4.7.3.36 __setUpSelectOutlineSprite()	116
4.7.3.37 __setUpSolarPVBuildOption()	116
4.7.3.38 __setUpTidalTurbineBuildOption()	117

4.7.3.39 __setUpTileExplosionReel()	117
4.7.3.40 __setUpTileSprite()	118
4.7.3.41 __setUpWaveEnergyConverterBuildOption()	118
4.7.3.42 __setUpWindTurbineBuildOption()	119
4.7.3.43 assess()	119
4.7.3.44 decorateTile()	120
4.7.3.45 draw()	121
4.7.3.46 processEvent()	122
4.7.3.47 processMessage()	122
4.7.3.48 setTileResource() [1/2]	123
4.7.3.49 setTileResource() [2/2]	124
4.7.3.50 setTileType() [1/2]	124
4.7.3.51 setTileType() [2/2]	125
4.7.3.52 toggleResourceOverlay()	125
4.7.4 Member Data Documentation	126
4.7.4.1 assets_manager_ptr	126
4.7.4.2 build_menu_backing	126
4.7.4.3 build_menu_backing_text	126
4.7.4.4 build_menu_open	126
4.7.4.5 build_menu_options_text_vec	126
4.7.4.6 build_menu_options_vec	127
4.7.4.7 credits	127
4.7.4.8 decoration_cleared	127
4.7.4.9 draw_explosion	127
4.7.4.10 event_ptr	127
4.7.4.11 explosion_frame	127
4.7.4.12 explosion_sprite_reel	128
4.7.4.13 frame	128
4.7.4.14 game_phase	128
4.7.4.15 has_improvement	128
4.7.4.16 is_selected	128
4.7.4.17 magnifying_glass_sprite	128
4.7.4.18 major_radius	129
4.7.4.19 message_hub_ptr	129
4.7.4.20 minor_radius	129
4.7.4.21 node_sprite	129
4.7.4.22 position_x	129
4.7.4.23 position_y	129
4.7.4.24 render_window_ptr	130
4.7.4.25 resource_assessed	130
4.7.4.26 resource_assessment	130
4.7.4.27 resource_chip_sprite	130

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

4.10.3.2	__handleMouseButtonEvents()	143
4.10.3.3	__setUpTileImprovementSpriteStatic()	144
4.10.3.4	draw()	144
4.10.3.5	getTileOptionsSubstring()	145
4.10.3.6	processEvent()	145
4.10.3.7	processMessage()	146
4.10.4	Member Data Documentation	146
4.10.4.1	smoke_da	146
4.10.4.2	smoke_dx	146
4.10.4.3	smoke_dy	146
4.10.4.4	smoke_prob	146
4.10.4.5	smoke_sprite_list	147
4.11	SolarPV Class Reference	147
4.11.1	Detailed Description	148
4.11.2	Constructor & Destructor Documentation	148
4.11.2.1	SolarPV()	148
4.11.2.2	~SolarPV()	149
4.11.3	Member Function Documentation	149
4.11.3.1	__handleKeyPressEvents()	150
4.11.3.2	__handleMouseButtonEvents()	150
4.11.3.3	__setUpTileImprovementSpriteStatic()	151
4.11.3.4	draw()	151
4.11.3.5	getTileOptionsSubstring()	151
4.11.3.6	processEvent()	152
4.11.3.7	processMessage()	152
4.12	TidalTurbine Class Reference	153
4.12.1	Detailed Description	154
4.12.2	Constructor & Destructor Documentation	154
4.12.2.1	TidalTurbine()	154
4.12.2.2	~TidalTurbine()	155
4.12.3	Member Function Documentation	155
4.12.3.1	__handleKeyPressEvents()	155
4.12.3.2	__handleMouseButtonEvents()	156
4.12.3.3	__setUpTileImprovementSpriteAnimated()	156
4.12.3.4	draw()	157
4.12.3.5	getTileOptionsSubstring()	157
4.12.3.6	processEvent()	158
4.12.3.7	processMessage()	158
4.13	TileImprovement Class Reference	158
4.13.1	Detailed Description	161
4.13.2	Constructor & Destructor Documentation	161
4.13.2.1	TileImprovement()	161

4.13.2.2 ~TileImprovement()	162
4.13.3 Member Function Documentation	162
4.13.3.1 __closeProductionMenu()	162
4.13.3.2 __handleKeyPressEvents()	163
4.13.3.3 __handleMouseButtonEvents()	163
4.13.3.4 __openProductionMenu()	164
4.13.3.5 __setUpProductionMenu()	164
4.13.3.6 draw()	164
4.13.3.7 getTileOptionsSubstring()	166
4.13.3.8 processEvent()	166
4.13.3.9 processMessage()	166
4.13.3.10 setIsSelected()	167
4.13.4 Member Data Documentation	167
4.13.4.1 assets_manager_ptr	167
4.13.4.2 credits	167
4.13.4.3 event_ptr	168
4.13.4.4 frame	168
4.13.4.5 game_phase	168
4.13.4.6 health	168
4.13.4.7 is_running	168
4.13.4.8 is_selected	168
4.13.4.9 just_built	169
4.13.4.10 message_hub_ptr	169
4.13.4.11 position_x	169
4.13.4.12 position_y	169
4.13.4.13 production_menu_backing	169
4.13.4.14 production_menu_backing_text	169
4.13.4.15 production_menu_open	170
4.13.4.16 render_window_ptr	170
4.13.4.17 tile_improvement_sprite_animated	170
4.13.4.18 tile_improvement_sprite_static	170
4.13.4.19 tile_improvement_string	170
4.13.4.20 tile_improvement_type	170
4.14 WaveEnergyConverter Class Reference	171
4.14.1 Detailed Description	172
4.14.2 Constructor & Destructor Documentation	172
4.14.2.1 WaveEnergyConverter()	172
4.14.2.2 ~WaveEnergyConverter()	173
4.14.3 Member Function Documentation	173
4.14.3.1 __handleKeyPressEvents()	173
4.14.3.2 __handleMouseButtonEvents()	174
4.14.3.3 __setUpTileImprovementSpriteAnimated()	174

4.14.3.4 draw()	175
4.14.3.5 getTileOptionsSubstring()	175
4.14.3.6 processEvent()	176
4.14.3.7 processMessage()	176
4.15 WindTurbine Class Reference	176
4.15.1 Detailed Description	177
4.15.2 Constructor & Destructor Documentation	178
4.15.2.1 WindTurbine()	178
4.15.2.2 ~WindTurbine()	178
4.15.3 Member Function Documentation	179
4.15.3.1 __handleKeyPressEvents()	179
4.15.3.2 __handleMouseButtonEvents()	179
4.15.3.3 __setUpTileImprovementSpriteAnimated()	180
4.15.3.4 draw()	180
4.15.3.5 getTileOptionsSubstring()	181
4.15.3.6 processEvent()	181
4.15.3.7 processMessage()	181
5 File Documentation	183
5.1 header/ContextMenu.h File Reference	183
5.1.1 Detailed Description	184
5.1.2 Enumeration Type Documentation	184
5.1.2.1 ConsoleState	184
5.2 header/DieselGenerator.h File Reference	184
5.2.1 Detailed Description	185
5.3 header/EnergyStorageSystem.h File Reference	185
5.3.1 Detailed Description	186
5.4 header/ESC_core/AssetsManager.h File Reference	186
5.4.1 Detailed Description	187
5.5 header/ESC_core/constants.h File Reference	187
5.5.1 Detailed Description	189
5.5.2 Function Documentation	190
5.5.2.1 FOREST_GREEN()	190
5.5.2.2 LAKE_BLUE()	190
5.5.2.3 MENU_FRAME_GREY()	190
5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()	190
5.5.2.5 MONOCHROME_TEXT_AMBER()	191
5.5.2.6 MONOCHROME_TEXT_GREEN()	191
5.5.2.7 MONOCHROME_TEXT_RED()	191
5.5.2.8 MOUNTAINS_GREY()	191
5.5.2.9 OCEAN_BLUE()	191
5.5.2.10 PLAINS_YELLOW()	192

5.5.2.11 RESOURCE_CHIP_GREY()	192
5.5.2.12 VISUAL_SCREEN_FRAME_GREY()	192
5.5.3 Variable Documentation	192
5.5.3.1 BUILD_SETTLEMENT_COST	192
5.5.3.2 CLEAR_FOREST_COST	192
5.5.3.3 CLEAR_MOUNTAINS_COST	193
5.5.3.4 CLEAR_PLAINS_COST	193
5.5.3.5 CO2E_KG_PER_LITRE_DIESEL	193
5.5.3.6 DIESEL_GENERATOR_BUILD_COST	193
5.5.3.7 EMISSIONS_LIFETIME_LIMIT_TONNES	193
5.5.3.8 ENERGY_STORAGE_SYSTEM_BUILD_COST	193
5.5.3.9 FLOAT_TOLERANCE	194
5.5.3.10 FRAMES_PER_SECOND	194
5.5.3.11 GAME_CHANNEL	194
5.5.3.12 GAME_HEIGHT	194
5.5.3.13 GAME_STATE_CHANNEL	194
5.5.3.14 GAME_WIDTH	194
5.5.3.15 HEX_MAP_CHANNEL	195
5.5.3.16 NO_TILE_SELECTED_CHANNEL	195
5.5.3.17 RESOURCE_ASSESSMENT_COST	195
5.5.3.18 SCRAP_COST	195
5.5.3.19 SECONDS_PER_FRAME	195
5.5.3.20 SECONDS_PER_MONTH	195
5.5.3.21 SECONDS_PER_YEAR	196
5.5.3.22 SOLAR_PV_BUILD_COST	196
5.5.3.23 SOLAR_PV_WATER_BUILD_MULTIPLIER	196
5.5.3.24 STARTING_CREDITS	196
5.5.3.25 STARTING_POPULATION	196
5.5.3.26 TIDAL_TURBINE_BUILD_COST	196
5.5.3.27 TILE_RESOURCE_CUMULATIVE_PROBABILITIES	197
5.5.3.28 TILE_SELECTED_CHANNEL	197
5.5.3.29 TILE_STATE_CHANNEL	197
5.5.3.30 TILE_TYPE_CUMULATIVE_PROBABILITIES	197
5.5.3.31 WAVE_ENERGY_CONVERTER_BUILD_COST	197
5.5.3.32 WIND_TURBINE_BUILD_COST	198
5.5.3.33 WIND_TURBINE_WATER_BUILD_MULTIPLIER	198
5.6 header/ESC_core/doxygen_cite.h File Reference	198
5.6.1 Detailed Description	198
5.7 header/ESC_core/includes.h File Reference	198
5.7.1 Detailed Description	199
5.8 header/ESC_core/MessageHub.h File Reference	199
5.8.1 Detailed Description	200

5.9 header/ESC_core/testing_utils.h File Reference	200
5.9.1 Detailed Description	201
5.9.2 Function Documentation	201
5.9.2.1 expectedErrorNotDetected()	201
5.9.2.2 printGold()	202
5.9.2.3 printGreen()	202
5.9.2.4 printRed()	202
5.9.2.5 testFloatEquals()	203
5.9.2.6 testGreaterThan()	203
5.9.2.7 testGreaterThanOrEqualTo()	204
5.9.2.8 testLessThan()	205
5.9.2.9 testLessThanOrEqualTo()	205
5.9.2.10 testTruth()	206
5.10 header/Game.h File Reference	206
5.10.1 Enumeration Type Documentation	207
5.10.1.1 GamePhase	207
5.11 header/HexMap.h File Reference	208
5.11.1 Detailed Description	209
5.12 header/HexTile.h File Reference	209
5.12.1 Detailed Description	210
5.12.2 Enumeration Type Documentation	210
5.12.2.1 TileResource	210
5.12.2.2 TileType	211
5.13 header/Settlement.h File Reference	211
5.13.1 Detailed Description	212
5.14 header/SolarPV.h File Reference	212
5.14.1 Detailed Description	213
5.15 header/TidalTurbine.h File Reference	213
5.15.1 Detailed Description	214
5.16 header/TileImprovement.h File Reference	214
5.16.1 Detailed Description	215
5.16.2 Enumeration Type Documentation	215
5.16.2.1 TileImprovementType	215
5.17 header/WaveEnergyConverter.h File Reference	216
5.17.1 Detailed Description	217
5.18 header/WindTurbine.h File Reference	217
5.18.1 Detailed Description	217
5.19 source/ContextMenu.cpp File Reference	218
5.19.1 Detailed Description	218
5.20 source/DieselGenerator.cpp File Reference	218
5.20.1 Detailed Description	218
5.21 source/EnergyStorageSystem.cpp File Reference	218

5.21.1 Detailed Description	219
5.22 source/ESC_core/AssetsManager.cpp File Reference	219
5.22.1 Detailed Description	219
5.23 source/ESC_core/MessageHub.cpp File Reference	219
5.23.1 Detailed Description	219
5.24 source/ESC_core/testing_utils.cpp File Reference	220
5.24.1 Detailed Description	220
5.24.2 Function Documentation	220
5.24.2.1 expectedErrorNotDetected()	220
5.24.2.2 printGold()	221
5.24.2.3 printGreen()	221
5.24.2.4 printRed()	221
5.24.2.5 testFloatEquals()	222
5.24.2.6 testGreaterThan()	222
5.24.2.7 testGreaterThanOrEqualTo()	223
5.24.2.8 testLessThan()	224
5.24.2.9 testLessThanOrEqualTo()	224
5.24.2.10 testTruth()	225
5.25 source/Game.cpp File Reference	226
5.25.1 Detailed Description	226
5.26 source/HexMap.cpp File Reference	226
5.26.1 Detailed Description	226
5.27 source/HexTile.cpp File Reference	227
5.27.1 Detailed Description	227
5.28 source/main.cpp File Reference	227
5.28.1 Detailed Description	227
5.28.2 Function Documentation	228
5.28.2.1 constructRenderWindow()	228
5.28.2.2 loadAssets()	228
5.28.2.3 main()	231
5.29 source/Settlement.cpp File Reference	231
5.29.1 Detailed Description	231
5.30 source/SolarPV.cpp File Reference	232
5.30.1 Detailed Description	232
5.31 source/TidalTurbine.cpp File Reference	232
5.31.1 Detailed Description	232
5.32 source/TileImprovement.cpp File Reference	232
5.32.1 Detailed Description	233
5.33 source/WaveEnergyConverter.cpp File Reference	233
5.33.1 Detailed Description	233
5.34 source/WindTurbine.cpp File Reference	233
5.34.1 Detailed Description	233

Bibliography	235
Index	237

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AssetsManager	7
ContextMenu	19
Game	51
HexMap	65
HexTile	88
Message	132
MessageHub	133
TileImprovement	158
DieselGenerator	37
EnergyStorageSystem	45
Settlement	140
SolarPV	147
TidalTurbine	153
WaveEnergyConverter	171
WindTurbine	176

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)	45
Game	A class which acts as the central class for the game, by containing all other classes and implementing the game loop	51
HexMap	A class which defines a hex map of hex tiles	65
HexTile	A class which defines a hex tile of the hex map	88
Message	A structure which defines a standard message format	132
MessageHub	A class which acts as a central hub for inter-object message traffic	133
Settlement	A settlement class (child class of TileImprovement)	140
SolarPV	A settlement class (child class of TileImprovement)	147
TidalTurbine	A settlement class (child class of TileImprovement)	153
TileImprovement	A base class for the tile improvement hierarchy	158
WaveEnergyConverter	A settlement class (child class of TileImprovement)	171
WindTurbine	A settlement class (child class of TileImprovement)	176

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	183
header/ DieselGenerator.h	
Header file for the DieselGenerator class	184
header/ EnergyStorageSystem.h	
Header file for the EnergyStorageSystem class	185
header/ Game.h	206
header/ HexMap.h	
Header file for the HexMap class	208
header/ HexTile.h	
Header file for the Game class	209
header/ Settlement.h	
Header file for the Settlement class	211
header/ SolarPV.h	
Header file for the SolarPV class	212
header/ TidalTurbine.h	
Header file for the TidalTurbine class	213
header/ TileImprovement.h	
Header file for the TileImprovement class	214
header/ WaveEnergyConverter.h	
Header file for the WaveEnergyConverter class	216
header/ WindTurbine.h	
Header file for the WindTurbine class	217
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	186
header/ESC_core/ constants.h	
Header file for various constants	187
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	198
header/ESC_core/ includes.h	
Header file for various includes	198
header/ESC_core/ MessageHub.h	
Header file for the MessageHub class	199
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	200

source/ ContextMenu.cpp	Implementation file for the ContextMenu class	218
source/ DieselGenerator.cpp	Implementation file for the DieselGenerator class	218
source/ EnergyStorageSystem.cpp	Implementation file for the EnergyStorageSystem class	218
source/ Game.cpp	Implementation file for the Game class	226
source/ HexMap.cpp	Implementation file for the HexMap class	226
source/ HexTile.cpp	Implementation file for the HexTile class	227
source/ main.cpp	Implementation file for main() for Road To Zero	227
source/ Settlement.cpp	Implementation file for the Settlement class	231
source/ SolarPV.cpp	Implementation file for the SolarPV class	232
source/ TidalTurbine.cpp	Implementation file for the TidalTurbine class	232
source/ TileImprovement.cpp	Implementation file for the TileImprovement class	232
source/ WaveEnergyConverter.cpp	Implementation file for the WaveEnergyConverter class	233
source/ WindTurbine.cpp	Implementation file for the WindTurbine class	233
source/ESC_core/ AssetsManager.cpp	Implementation file for the AssetsManager class	219
source/ESC_core/ MessageHub.cpp	Implementation file for the MessageHub class	219
source/ESC_core/ testing_utils.cpp	Implementation file for various testing utilities	220

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.

```
142 {
143     //...
144
145     std::cout << "AssetsManager constructed at " << this << std::endl;
146
147     return;
148 } /* AssetsManager() */
```

4.1.2.2 ~AssetsManager()

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

Destructor for the [AssetsManager](#) class.

```
771 {
772     this->clear();
773
774     std::cout << "AssetsManager at " << this << " destroyed" << std::endl;
775
776     return;
777 } /* ~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).

```
79 {
80     // 1. check key, throw error if already in use
81     if (this->soundbuffer_map.count(sound_key) > 0) {
82         std::string error_str = "ERROR AssetsManager::__loadSoundBuffer() sound key ";
83         error_str += sound_key;
84         error_str += " is already in use";
85
86         this->clear();
87
88         #ifdef _WIN32
89             std::cout << error_str << std::endl;
90         #endif /* _WIN32 */
91
92         throw std::runtime_error(error_str);
93     }
94
95
96     // 2. load from file, throw error on fail
97     sf::SoundBuffer* soundbuffer_ptr = new sf::SoundBuffer();
98
99     if (not soundbuffer_ptr->loadFromFile(path_2_sound)) {
100         std::string error_str = "ERROR AssetsManager::__loadSoundBuffer() could not load ";
101         error_str += "soundbuffer at ";
102         error_str += path_2_sound;
103
104         this->clear();
105
106         #ifdef _WIN32
107             std::cout << error_str << std::endl;
108         #endif /* _WIN32 */
109
110         throw std::runtime_error(error_str);
111     }
112
113 }
```

```

114 // 3. insert into soundbuffer map
115 this->soundbuffer_map.insert(
116     std::pair<std::string, sf::SoundBuffer*>(sound_key, soundbuffer_ptr)
117 );
118
119 std::cout << "SoundBuffer " << sound_key << " inserted into soundbuffer map" <<
120     std::endl;
121
122 return;
123 } /* __loadSoundBuffer() */

```

4.1.3.2 clear()

```

void AssetsManager::clear (
    void )

```

Method to clear all loaded assets.

```

678 {
679     // 1. clear fonts
680     std::map<std::string, sf::Font*>::iterator font_iter;
681     for (
682         font_iter = this->font_map.begin();
683         font_iter != this->font_map.end();
684         font_iter++
685     ) {
686         delete font_iter->second;
687
688         std::cout << "Font " << font_iter->first << " deleted from font map" <<
689             std::endl;
690     }
691     this->font_map.clear();
692
693     // 2. clear textures
694     std::map<std::string, sf::Texture*>::iterator texture_iter;
695     for (
696         texture_iter = this->texture_map.begin();
697         texture_iter != this->texture_map.end();
698         texture_iter++
699     ) {
700         delete texture_iter->second;
701
702         std::cout << "Texture " << texture_iter->first << " deleted from texture map" <<
703             std::endl;
704     }
705     this->texture_map.clear();
706
707     // 3. clear sound buffers
708     std::map<std::string, sf::SoundBuffer*>::iterator soundbuffer_iter;
709     for (
710         soundbuffer_iter = this->soundbuffer_map.begin();
711         soundbuffer_iter != this->soundbuffer_map.end();
712         soundbuffer_iter++
713     ) {
714         delete soundbuffer_iter->second;
715
716         std::cout << "SoundBuffer " << soundbuffer_iter->first <<
717             " deleted from soundbuffer map" << std::endl;
718     }
719     this->soundbuffer_map.clear();
720
721     // 4. clear sounds
722     std::map<std::string, sf::Sound*>::iterator sound_iter;
723     for (
724         sound_iter = this->sound_map.begin();
725         sound_iter != this->sound_map.end();
726         sound_iter++
727     ) {
728         sound_iter->second->stop();
729         delete sound_iter->second;
730
731         std::cout << "Sound " << sound_iter->first << " deleted from sound map" <<
732             std::endl;
733     }
734     this->sound_map.clear();
735
736 }
737
738

```

```

739
740 // 5. clear tracks
741 std::map<std::string, sf::Music*>::iterator track_iter;
742 for (
743     track_iter = this->track_map.begin();
744     track_iter != this->track_map.end();
745     track_iter++)
746 {
747     track_iter->second->stop();
748     delete track_iter->second;
749
750     std::cout << "Track " << track_iter->first << " deleted from track map" <<
751         std::endl;
752 }
753 this->track_map.clear();
754
755 return;
756 } /* 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.

```

642 {
643     return this->current_track->first;
644 } /* 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.

```

383 {
384     // 1. check key, throw error if not found
385     if (this->font_map.count(font_key) <= 0) {
386         std::string error_str = "ERROR AssetsManager::getFont() font key ";
387         error_str += font_key;
388         error_str += " is not contained in font map";
389
390         this->clear();
391
392         #ifdef _WIN32

```

```

393         std::cout << error_str << std::endl;
394     #endif /* _WIN32 */
395
396     throw std::runtime_error(error_str);
397 }
398
399 return this->font_map[font_key];
400 } /* 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.

```

493 {
494     // 1. check key, throw error if not found
495     if (this->sound_map.count(sound_key) <= 0) {
496         std::string error_str = "ERROR AssetsManager::getSound() sound key ";
497         error_str += sound_key;
498         error_str += " is not contained in sound map";
499
500         this->clear();
501
502         #ifdef _WIN32
503             std::cout << error_str << std::endl;
504         #endif /* _WIN32 */
505
506         throw std::runtime_error(error_str);
507     }
508
509     return this->sound_map[sound_key];
510 } /* 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.

```

457 {
458     // 1. check key, throw error if not found
459     if (this->soundbuffer_map.count(sound_key) <= 0) {
460         std::string error_str = "ERROR AssetsManager::getSoundBuffer() sound key ";
461         error_str += sound_key;
462         error_str += " is not contained in soundbuffer map";
463
464         this->clear();
465
466         #ifdef _WIN32
467             std::cout << error_str << std::endl;
468         #endif /* _WIN32 */
469
470         throw std::runtime_error(error_str);
471     }
472
473     return this->soundbuffer_map[sound_key];
474 } /* 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.

```

420 {
421     // 1. check key, throw error if not found
422     if (this->texture_map.count(texture_key) <= 0) {
423         std::string error_str = "ERROR AssetsManager::getTexture() texture key ";
424         error_str += texture_key;
425         error_str += " is not contained in texture map";
426
427         this->clear();
428
429         #ifdef _WIN32
430             std::cout << error_str << std::endl;
431         #endif /* _WIN32 */
432
433         throw std::runtime_error(error_str);
434     }
435
436     return this->texture_map[texture_key];
437 } /* 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.

```
661 {
662     return this->current_track->second->getStatus();
663 } /* 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).

```
167 {
168     // 1. check key, throw error if already in use
169     if (this->font_map.count(font_key) > 0) {
170         std::string error_str = "ERROR AssetsManager::loadFont() font key ";
171         error_str += font_key;
172         error_str += " is already in use";
173
174         this->clear();
175
176         #ifdef _WIN32
177             std::cout << error_str << std::endl;
178         #endif /* _WIN32 */
179
180         throw std::runtime_error(error_str);
181     }
182
183
184     // 2. load from file, throw error on fail
185     sf::Font* font_ptr = new sf::Font();
186
187     if (not font_ptr->loadFromFile(path_2_font)) {
188         std::string error_str = "ERROR AssetsManager::loadFont() could not load ";
189         error_str += "font at ";
190         error_str += path_2_font;
191
192         this->clear();
193
194         #ifdef _WIN32
195             std::cout << error_str << std::endl;
196         #endif /* _WIN32 */
197
198         throw std::runtime_error(error_str);
199     }
200
201
202     // 3. insert into font map
203     this->font_map.insert(std::pair<std::string, sf::Font*>(font_key, font_ptr));
204
205     std::cout << "Font " << font_key << " inserted into font map" << std::endl;
206
207     return;
208 } /* 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).

```
291 {
292     // 1. create an associated sf::SoundBuffer
293     this->__loadSoundBuffer(path_2_sound, sound_key);
294
295     // 2. associate sf::Sound with sf::SoundBuffer
296     sf::Sound* sound_ptr = new sf::Sound();
297     sound_ptr->setBuffer(*(this->soundbuffer_map[sound_key]));
298
299     // 3. insert into sound map
300     this->sound_map.insert(std::pair<std::string, sf::Sound*>(sound_key, sound_ptr));
301
302     std::cout << "Sound " << sound_key << " inserted into sound map" << std::endl;
303
304     return;
305 } /* 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).

```
228 {
229     // 1. check key, throw error if already in use
230     if (this->texture_map.count(texture_key) > 0) {
231         std::string error_str = "ERROR AssetsManager::loadTexture() texture key ";
232         error_str += texture_key;
233         error_str += " is already in use";
234
235         this->clear();
236
237         #ifdef _WIN32
238             std::cout << error_str << std::endl;
239         #endif /* _WIN32 */
240
241         throw std::runtime_error(error_str);
242     }
243
244     // 2. load from file, throw error on fail
245     sf::Texture* texture_ptr = new sf::Texture();
246
247     if (not texture_ptr->loadFromFile(path_2_texture)) {
248         std::string error_str = "ERROR AssetsManager::loadTexture() could not load ";
249         error_str += "texture at ";
250         error_str += path_2_texture;
251
252         this->clear();
253
254         #ifdef _WIN32
255             std::cout << error_str << std::endl;
256         #endif
```

```

257         #endif /* _WIN32 */
258
259         throw std::runtime_error(error_str);
260     }
261
262
263     // 3. insert into texture map
264     this->texture_map.insert(
265         std::pair<std::string, sf::Texture*>(texture_key, texture_ptr)
266     );
267
268     std::cout << "Texture " << texture_key << " inserted into texture map" << std::endl;
269
270     return;
271 } /* 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).

```

324 {
325     // 1. check key, throw error if already in use
326     if (this->track_map.count(track_key) > 0) {
327         std::string error_str = "ERROR AssetsManager::loadTrack() track key ";
328         error_str += track_key;
329         error_str += " is already in use";
330
331         this->clear();
332
333         #ifdef _WIN32
334             std::cout << error_str << std::endl;
335         #endif /* _WIN32 */
336
337         throw std::runtime_error(error_str);
338     }
339
340     // 2. open from file, throw error on fail
341     sf::Music* track_ptr = new sf::Music();
342
343     if (not track_ptr->openFromFile(path_2_track)) {
344         std::string error_str = "ERROR AssetsManager::loadTrack() could not open ";
345         error_str += "track at ";
346         error_str += path_2_track;
347
348         this->clear();
349
350         #ifdef _WIN32
351             std::cout << error_str << std::endl;
352         #endif /* _WIN32 */
353
354         throw std::runtime_error(error_str);
355     }
356
357     // 3. insert into track map
358     this->track_map.insert(std::pair<std::string, sf::Music*>(track_key, track_ptr));
359     this->current_track = this->track_map.begin();
360
361     std::cout << "Track " << track_key << " inserted into track map" << std::endl;
362
363     return;
364 } /* 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.

```
583 {
584     // 1. stop current track
585     this->stopTrack();
586
587     // 2. increment current track
588     this->current_track++;
589
590     // 3. handle wrap around
591     if (this->current_track == this->track_map.end()) {
592         this->current_track = this->track_map.begin();
593     }
594
595     return;
596 } /* nextTrack() */
```

4.1.3.14 pauseTrack()

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

Method to pause the current track.

```
544 {
545     this->current_track->second->pause();
546
547     return;
548 } /* pauseTrack() */
```

4.1.3.15 playTrack()

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

Method to play the current track.

```
525 {
526     this->current_track->second->play();
527
528     return;
529 } /* 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.

```
612 {
613     // 1. stop current track
614     this->stopTrack();
615
616     // 2. handle wrap around
617     if (this->current_track == this->track_map.begin()) {
618         this->current_track = this->track_map.end();
619     }
620
621     // 3. decrement current track
622     this->current_track--;
623
624     return;
625 } /* previousTrack() */
```

4.1.3.17 stopTrack()

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

Method to stop the current track.

```
563 {
564     this->current_track->second->stop();
565
566     return;
567 } /* 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:

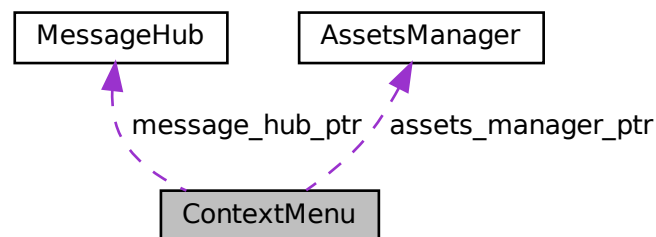
- header/ESC_core/[AssetsManager.h](#)
- source/ESC_core/[AssetsManager.cpp](#)

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.
- unsigned long long 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.

```

849 {
850     // 1. set attributes
851
852     // 1.1. private
853     this->event_ptr = event_ptr;
854     this->render_window_ptr = render_window_ptr;
855
856     this->assets_manager_ptr = assets_manager_ptr;
857     this->message_hub_ptr = message_hub_ptr;
858
859     // 1.2. public
860     this->console_state = ConsoleState :: NONE_STATE;
861     this->__setConsoleState(ConsoleState :: READY);
862
863     this->console_string_changed = true;
864     this->game_menu_up = false;
865
866     this->frame = 0;
867
868     this->position_x = GAME_WIDTH;
869     this->position_y = 0;
870
871     // 2. set up and position drawable attributes
872     this->__setUpMenuFrame();
873     this->__setUpVisualScreen();
874     this->__setUpVisualScreenFrame();
875     this->__setUpConsoleScreen();
876     this->__setUpConsoleScreenFrame();
877
878     std::cout << "ContextMenu constructed at " << this << std::endl;
879
880     return;
881 } /* ContextMenu() */

```

4.2.2.2 ~ContextMenu()

```

ContextMenu::~ContextMenu (
    void )

```

Destructor for the [ContextMenu](#) class.

```

1031 {
1032     std::cout << "ContextMenu at " << this << " destroyed" << std::endl;
1033
1034     return;
1035 } /* ~ContextMenu() */

```

4.2.3 Member Function Documentation

4.2.3.1 __drawConsoleScreenFrame()

```

void ContextMenu::__drawConsoleScreenFrame (
    void ) [private]

```

Helper method to draw console screen frame.


```

467 {
468     this->render_window_ptr->draw(this->console_screen_frame_top);
469     this->render_window_ptr->draw(this->console_screen_frame_left);
470     this->render_window_ptr->draw(this->console_screen_frame_bottom);
471     this->render_window_ptr->draw(this->console_screen_frame_right);
472
473     return;
474 } /* __drawContextScreenFrame() */

```

4.2.3.2 __drawConsoleText()

```

void ContextMenu::__drawConsoleText (
    void ) [private]

```

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

```

590 {
591     // 1. set up console text (drawable)
592     sf::Text console_text;
593
594     if (this->console_string_changed) {
595         this->assets_manager_ptr->getSound("console string print")->play();
596
597         console_text.setString(this->console_string.substr(0, this->console_substring_idx));
598
599         this->console_substring_idx++;
600
601         while (
602             (this->console_string.substr(0, this->console_substring_idx).back() == ' ') or
603             (this->console_string.substr(0, this->console_substring_idx).back() == '\n')
604         ) {
605             this->console_substring_idx++;
606
607             if (this->console_substring_idx >= this->console_string.size()) {
608                 break;
609             }
610         }
611
612         if (this->console_substring_idx >= this->console_string.size()) {
613             this->console_string_changed = false;
614         }
615     }
616
617     else {
618         console_text.setString(this->console_string);
619     }
620
621     console_text.setFont(*(this->assets_manager_ptr->getFont("Glass_TTY_VT220")));
622     console_text.setCharacterSize(16);
623     console_text.setFillColor(MONOCROME_TEXT_GREEN);
624
625     console_text.setPosition(
626         this->position_x - 50 - 300 + 16,
627         this->position_y + GAME_HEIGHT - 50 - 340 + 16
628     );
629
630
631     // 2. draw console text
632     this->render_window_ptr->draw(console_text);
633
634
635     // 3. assemble and draw blinking console cursor
636     if ((this->frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
637         sf::RectangleShape console_cursor(sf::Vector2f(10, 16));
638
639         console_cursor.setFillColor(MONOCROME_TEXT_GREEN);
640
641         console_cursor.setPosition(
642             console_text.getPosition().x,
643             console_text.getPosition().y + console_text.getLocalBounds().height + 10
644         );
645
646         this->render_window_ptr->draw(console_cursor);
647     }
648
649     // 4. updating frame count if console is in menu state
650     if (this->console_state == ConsoleState::MENU) {
651         std::string frame_count_string = "FRAME: ";
652         frame_count_string += std::to_string(this->frame);

```

```

653
654     sf::Text frame_count_text(
655         frame_count_string,
656         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
657         16
658     );
659
660     frame_count_text.setFillColor(MONOCROME_TEXT_GREEN);
661
662     frame_count_text.setPosition(
663         console_text.getPosition().x,
664         console_text.getPosition().y + console_text.getLocalBounds().height - 10
665     );
666
667     this->render_window_ptr->draw(frame_count_text);
668 }
669
670 return;
671 } /* __drawConsoleText() */

```

4.2.3.3 __drawVisualScreenFrame()

```

void ContextMenu::__drawVisualScreenFrame (
    void ) [private]

```

Helper method to draw visual screen frame.

```

242 {
243     this->render_window_ptr->draw(this->visual_screen_frame_top);
244     this->render_window_ptr->draw(this->visual_screen_frame_left);
245     this->render_window_ptr->draw(this->visual_screen_frame_bottom);
246     this->render_window_ptr->draw(this->visual_screen_frame_right);
247
248     return;
249 } /* __drawVisualScreenFrame() */

```

4.2.3.4 __handleKeyPressEvents()

```

void ContextMenu::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

686 {
687     switch (this->event_ptr->key.code) {
688         case (sf::Keyboard::Escape): {
689             if (this->console_state == ConsoleState :: MENU) {
690                 this->__setConsoleState(ConsoleState :: READY);
691             }
692
693             else {
694                 this->__setConsoleState(ConsoleState :: MENU);
695             }
696
697             break;
698         }
699
700         case (sf::Keyboard::Q): {
701             if (this->console_state == ConsoleState :: MENU) {
702                 this->__sendQuitGameMessage();
703             }
704         }
705
706         case (sf::Keyboard::R): {
707             if (this->console_state == ConsoleState :: MENU) {
708                 this->__sendRestartGameMessage();
709             }
710         }
711     }
712 }
713

```

```

714
715         default: {
716             // do nothing!
717
718             break;
719         }
720     }
721
722     return;
723 } /* __handleKeyPressEvents() */

```

4.2.3.5 __handleMouseButtonEvents()

```

void ContextMenu::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

738 {
739     switch (this->event_ptr->mouseButton.button) {
740         case (sf::Mouse::Left): {
741             //...
742
743             break;
744         }
745
746         case (sf::Mouse::Right): {
747             //...
748
749             break;
750         }
751     }
752
753     default: {
754         // do nothing!
755
756         break;
757     }
758 }
759
760
761     return;
762 } /* __handleMouseButtonEvents() */

```

4.2.3.6 __sendQuitGameMessage()

```

void ContextMenu::__sendQuitGameMessage (
    void ) [private]

```

Helper method to format and send a quit game message.

```

777 {
778     Message quit_game_message;
779
780     quit_game_message.channel = GAME_CHANNEL;
781     quit_game_message.subject = "quit game";
782
783     this->message_hub_ptr->sendMessage(quit_game_message);
784
785     std::cout << "Quit game message sent by " << this << std::endl;
786     return;
787 } /* __sendQuitGameMessage() */

```

4.2.3.7 __sendRestartGameMessage()

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

Helper method to format and send a restart game message.

```
802 {
803     Message restart_game_message;
804
805     restart_game_message.channel = GAME_CHANNEL;
806     restart_game_message.subject = "restart game";
807
808     this->message_hub_ptr->sendMessage(restart_game_message);
809
810     std::cout << "Restart game message sent by " << this << std::endl;
811     return;
812 } /* __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.
----------------------	---

```
491 {
492     // 1. if no change, do nothing
493     if (this->console_state == console_state) {
494         return;
495     }
496
497     // 2. update console state, set console string accordingly
498     this->console_state = console_state;
499     this->__setConsoleString();
500
501     return;
502 } /* __setConsoleState() */
```

4.2.3.9 __setConsoleString()

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

Helper method to set console string depending on console state.

```
517 {
518     this->console_string_changed = true;
519     this->console_substring_idx = 0;
520
521     this->console_string.clear();
522
523     switch (this->console_state) {
524     case (ConsoleState :: MENU): {
525         // 32 char x 17 line console "-----\n";
526         this->console_string = "          **** MENU ****          \n";
527         this->console_string += "          \n";
528         this->console_string += "[R]:  RESTART          \n";
529         this->console_string += "          \n";
530         this->console_string += "[TAB]: TOGGLE RESOURCE OVERLAY \n";
531     }
```

```

531         this->console_string += "[T]:  TUTORIAL          \n";
532         this->console_string += "                  \n";
533         this->console_string += "                  \n";
534         this->console_string += "                  \n";
535         this->console_string += "                  \n";
536         this->console_string += "                  \n";
537         this->console_string += "                  \n";
538         this->console_string += "                  \n";
539         this->console_string += "[Q]:    QUIT          \n";
540         this->console_string += "[ESC]:  CLOSE MENU    \n";
541         this->console_string += "                  \n";
542
543         break;
544     }
545
546     case (ConsoleState :: TILE): {
547         // take console string from tile state message
548
549         break;
550     }
551
552
553
554     default: {
555         //          32 char x 17 line console "-----\n";
556         this->console_string = "    **** RTZ 64 CONTEXT V12 **** \n";
557         this->console_string += "                  \n";
558         this->console_string += "64K RAM SYSTEM  38911 BYTES FREE\n";
559         this->console_string += "                  \n";
560         this->console_string += "[TAB]:  TOGGLE RESOURCE OVERLAY \n";
561         this->console_string += "                  \n";
562         this->console_string += "[ESC]:           MENU          \n";
563         this->console_string += "[LEFT CLICK]:  TILE INFO/OPTIONS\n";
564         this->console_string += "[RIGHT CLICK]: CLEAR SELECTION \n";
565         this->console_string += "                  \n";
566         this->console_string += "[ENTER]:  END TURN            \n";
567         this->console_string += "                  \n";
568         this->console_string += "READY.                        ";
569
570         break;
571     }
572 }
573
574 return;
575 } /* __setConsoleString() */

```

4.2.3.10 __setUpConsoleScreen()

```

void ContextMenu::__setUpConsoleScreen (
    void ) [private]

```

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

```

264 {
265     this->console_screen.setSize(sf::Vector2f(300, 340));
266     this->console_screen.setOrigin(300, 340);
267     this->console_screen.setPosition(
268         this->position_x - 50,
269         this->position_y + GAME_HEIGHT - 50
270     );
271     this->console_screen.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
272
273     return;
274 } /* __setUpConsoleScreen() */

```

4.2.3.11 __setUpConsoleScreenFrame()

```

void ContextMenu::__setUpConsoleScreenFrame (
    void ) [private]

```

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

```

289 {
290     int n_points = 4;
291
292     // 1. top framing
293     this->console_screen_frame_top.setPointCount(n_points);
294
295     this->console_screen_frame_top.setPoint(
296         0,
297         sf::Vector2f(
298             this->position_x - 50,
299             this->position_y + GAME_HEIGHT - 50 - 340
300         )
301     );
302     this->console_screen_frame_top.setPoint(
303         1,
304         sf::Vector2f(
305             this->position_x - 50 + 16,
306             this->position_y + GAME_HEIGHT - 50 - 340 - 16
307         )
308     );
309     this->console_screen_frame_top.setPoint(
310         2,
311         sf::Vector2f(
312             this->position_x - 350 - 16,
313             this->position_y + GAME_HEIGHT - 50 - 340 - 16
314         )
315     );
316     this->console_screen_frame_top.setPoint(
317         3,
318         sf::Vector2f(
319             this->position_x - 350,
320             this->position_y + GAME_HEIGHT - 50 - 340
321         )
322     );
323
324     this->console_screen_frame_top.setFillColors(VISUAL_SCREEN_FRAME_GREY);
325
326     this->console_screen_frame_top.setOutlineThickness(2);
327     this->console_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
328
329     this->console_screen_frame_top.move(0, -2);
330
331
332     // 2. left framing
333     this->console_screen_frame_left.setPointCount(n_points);
334
335     this->console_screen_frame_left.setPoint(
336         0,
337         sf::Vector2f(
338             this->position_x - 350,
339             this->position_y + GAME_HEIGHT - 50 - 340
340         )
341     );
342     this->console_screen_frame_left.setPoint(
343         1,
344         sf::Vector2f(
345             this->position_x - 350 - 16,
346             this->position_y + GAME_HEIGHT - 50 - 340 - 16
347         )
348     );
349     this->console_screen_frame_left.setPoint(
350         2,
351         sf::Vector2f(
352             this->position_x - 350 - 16,
353             this->position_y + GAME_HEIGHT - 50 + 16
354         )
355     );
356     this->console_screen_frame_left.setPoint(
357         3,
358         sf::Vector2f(
359             this->position_x - 350,
360             this->position_y + GAME_HEIGHT - 50
361         )
362     );
363
364     this->console_screen_frame_left.setFillColors(VISUAL_SCREEN_FRAME_GREY);
365
366     this->console_screen_frame_left.setOutlineThickness(2);
367     this->console_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
368
369     this->console_screen_frame_left.move(-2, 0);
370
371
372     // 3. bottom framing
373     this->console_screen_frame_bottom.setPointCount(n_points);
374

```

```

375     this->console_screen_frame_bottom.setPoint(
376         0,
377         sf::Vector2f(
378             this->position_x - 350,
379             this->position_y + GAME_HEIGHT - 50
380         )
381     );
382     this->console_screen_frame_bottom.setPoint(
383         1,
384         sf::Vector2f(
385             this->position_x - 350 - 16,
386             this->position_y + GAME_HEIGHT - 50 + 16
387         )
388     );
389     this->console_screen_frame_bottom.setPoint(
390         2,
391         sf::Vector2f(
392             this->position_x - 50 + 16,
393             this->position_y + GAME_HEIGHT - 50 + 16
394         )
395     );
396     this->console_screen_frame_bottom.setPoint(
397         3,
398         sf::Vector2f(
399             this->position_x - 50,
400             this->position_y + GAME_HEIGHT - 50
401         )
402     );
403
404     this->console_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
405
406     this->console_screen_frame_bottom.setOutlineThickness(2);
407     this->console_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
408
409     this->console_screen_frame_bottom.move(0, 2);
410
411     // 4. right framing
412     this->console_screen_frame_right.setPointCount(n_points);
413
414     this->console_screen_frame_right.setPoint(
415         0,
416         sf::Vector2f(
417             this->position_x - 50,
418             this->position_y + GAME_HEIGHT - 50
419         )
420     );
421
422     this->console_screen_frame_right.setPoint(
423         1,
424         sf::Vector2f(
425             this->position_x - 50 + 16,
426             this->position_y + GAME_HEIGHT - 50 + 16
427         )
428     );
429     this->console_screen_frame_right.setPoint(
430         2,
431         sf::Vector2f(
432             this->position_x - 50 + 16,
433             this->position_y + GAME_HEIGHT - 50 - 340 - 16
434         )
435     );
436     this->console_screen_frame_right.setPoint(
437         3,
438         sf::Vector2f(
439             this->position_x - 50,
440             this->position_y + GAME_HEIGHT - 50 - 340
441         )
442     );
443
444     this->console_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
445
446     this->console_screen_frame_right.setOutlineThickness(2);
447     this->console_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
448
449     this->console_screen_frame_right.move(2, 0);
450
451     return;
452 } /* __setUpConsoleScreenFrame() */

```

4.2.3.12 __setUpMenuFrame()

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

```
void ) [private]
```

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

```
68 {
69     this->menu_frame.setSize(sf::Vector2f(400, GAME_HEIGHT));
70     this->menu_frame.setOrigin(400, 0);
71     this->menu_frame.setPosition(this->position_x, this->position_y);
72     this->menu_frame.setFillColor(MENU_FRAME_GREY);
73
74     return;
75 } /* __setUpMenuFrame() */
```

4.2.3.13 __setUpVisualScreen()

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

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

```
90 {
91     this->visual_screen.setSize(sf::Vector2f(300, 300));
92     this->visual_screen.setOrigin(300, 0);
93     this->visual_screen.setPosition(this->position_x - 50, this->position_y + 50);
94     this->visual_screen.setFillColor(MONochrome_SCREEN_BACKGROUND);
95
96     return;
97 } /* __setUpVisualScreen() */
```

4.2.3.14 __setUpVisualScreenFrame()

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

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

```
112 {
113     int n_points = 4;
114
115     // 1. top framing
116     this->visual_screen_frame_top.setPointCount(n_points);
117
118     this->visual_screen_frame_top.setPoint(
119         0,
120         sf::Vector2f(this->position_x - 50, this->position_y + 50)
121     );
122     this->visual_screen_frame_top.setPoint(
123         1,
124         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
125     );
126     this->visual_screen_frame_top.setPoint(
127         2,
128         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
129     );
130     this->visual_screen_frame_top.setPoint(
131         3,
132         sf::Vector2f(this->position_x - 350, this->position_y + 50)
133     );
134
135     this->visual_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
136
137     this->visual_screen_frame_top.setOutlineThickness(2);
138     this->visual_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
139
140     this->visual_screen_frame_top.move(0, -2);
141
142
143     // 2. left framing
144     this->visual_screen_frame_left.setPointCount(n_points);
145
146     this->visual_screen_frame_left.setPoint(
```



```

147         0,
148         sf::Vector2f(this->position_x - 350, this->position_y + 50)
149     );
150     this->visual_screen_frame_left.setPoint(
151         1,
152         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
153     );
154     this->visual_screen_frame_left.setPoint(
155         2,
156         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
157     );
158     this->visual_screen_frame_left.setPoint(
159         3,
160         sf::Vector2f(this->position_x - 350, this->position_y + 350)
161     );
162
163     this->visual_screen_frame_left.setFillColor(VISUAL_SCREEN_FRAME_GREY);
164
165     this->visual_screen_frame_left.setOutlineThickness(2);
166     this->visual_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
167
168     this->visual_screen_frame_left.move(-2, 0);
169
170
171     // 3. bottom framing
172     this->visual_screen_frame_bottom.setPointCount(n_points);
173
174     this->visual_screen_frame_bottom.setPoint(
175         0,
176         sf::Vector2f(this->position_x - 350, this->position_y + 350)
177     );
178     this->visual_screen_frame_bottom.setPoint(
179         1,
180         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
181     );
182     this->visual_screen_frame_bottom.setPoint(
183         2,
184         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
185     );
186     this->visual_screen_frame_bottom.setPoint(
187         3,
188         sf::Vector2f(this->position_x - 50, this->position_y + 350)
189     );
190
191     this->visual_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
192
193     this->visual_screen_frame_bottom.setOutlineThickness(2);
194     this->visual_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
195
196     this->visual_screen_frame_bottom.move(0, 2);
197
198
199     // 4. right framing
200     this->visual_screen_frame_right.setPointCount(n_points);
201
202     this->visual_screen_frame_right.setPoint(
203         0,
204         sf::Vector2f(this->position_x - 50, this->position_y + 350)
205     );
206     this->visual_screen_frame_right.setPoint(
207         1,
208         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
209     );
210     this->visual_screen_frame_right.setPoint(
211         2,
212         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
213     );
214     this->visual_screen_frame_right.setPoint(
215         3,
216         sf::Vector2f(this->position_x - 50, this->position_y + 50)
217     );
218
219     this->visual_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
220
221     this->visual_screen_frame_right.setOutlineThickness(2);
222     this->visual_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
223
224     this->visual_screen_frame_right.move(2, 0);
225
226     return;
227 } /* __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.

```
1001 {
1002     // 1. menu frame
1003     this->render_window_ptr->draw(this->menu_frame);
1004
1005     // 2. visual screen
1006     this->render_window_ptr->draw(this->visual_screen);
1007     this->__drawVisualScreenFrame();
1008
1009     // 3. console screen
1010     this->render_window_ptr->draw(this->console_screen);
1011     this->__drawConsoleScreenFrame();
1012     this->__drawConsoleText();
1013
1014     this->frame++;
1015     return;
1016 } /* draw() */
```

4.2.3.16 processEvent()

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

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

```
896 {
897     if (this->event_ptr->type == sf::Event::KeyPressed) {
898         this->__handleKeyPressEvents();
899     }
900
901     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
902         this->__handleMouseButtonEvents();
903     }
904
905     return;
906 } /* processEvent() */
```

4.2.3.17 processMessage()

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

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

```
921 {
922     switch (this->console_state) {
923         case (ConsoleState :: TILE): {
924             // process no tile selected
925             if (not this->message_hub_ptr->isEmpty(NO_TILE_SELECTED_CHANNEL)) {
926                 Message no_tile_selected_message = this->message_hub_ptr->receiveMessage(
927                     NO_TILE_SELECTED_CHANNEL
928                 );
929
930                 if (no_tile_selected_message.subject == "no tile selected") {
931                     this->__setConsoleState(ConsoleState :: READY);
932
933                     std::cout << "No tile selected message received by " << this <<
934                         std::endl;
935                     this->message_hub_ptr->popMessage(NO_TILE_SELECTED_CHANNEL);
936                 }
937             }
938
939             // process tile state
```

```

940         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
941             Message tile_state_message = this->message_hub_ptr->receiveMessage(
942                 TILE_STATE_CHANNEL
943             );
944
945             if (tile_state_message.subject == "tile state") {
946                 this->console_string = tile_state_message.string_payload["console string"];
947
948                 this->console_string_changed = true;
949                 this->console_substring_idx = 0;
950
951                 std::cout << "Tile state message received by " << this << std::endl;
952                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
953             }
954         }
955
956         // process tile selected (subsequent left clicks causing program to hang)
957         if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
958             this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
959         }
960
961         break;
962     }
963
964     default: {
965         // process tile selected
966         if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
967             Message tile_selected_message = this->message_hub_ptr->receiveMessage(
968                 TILE_SELECTED_CHANNEL
969             );
970
971             if (tile_selected_message.subject == "tile selected") {
972                 this->__setConsoleState(ConsoleState :: TILE);
973
974                 std::cout << "Tile selected message received by " << this <<
975                     std::endl;
976                 this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
977             }
978         }
979
980         break;
981     }
982 }
983
984 return;
985 } /* 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

```
unsigned long long 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:

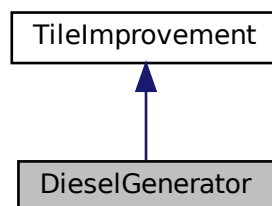
- header/[ContextMenu.h](#)
- source/[ContextMenu.cpp](#)

4.3 DieselGenerator Class Reference

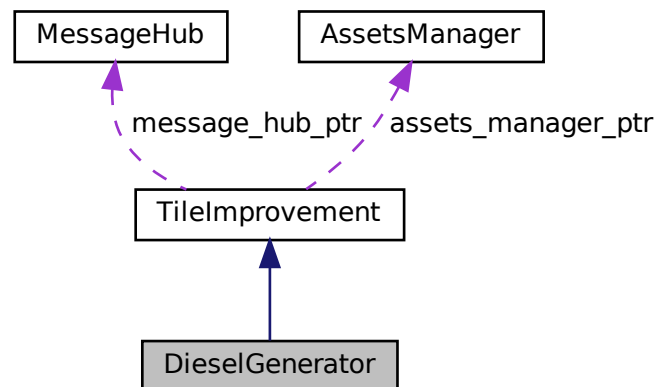
A settlement class (child class of [TileImprovement](#)).

```
#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.
- [std::string getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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

- int [capacity_kW](#)
The rated production capacity [kW] of the diesel generator.
- int [production_MWh](#)
The current production [MWh] of the diesel generator.
- int [max_production_MWh](#)
The maximum production [MWh] for this turn.
- 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.

```
214 :
215 TileImprovement (
216     position_x,
217     position_y,
218     event_ptr,
219     render_window_ptr,
220     assets_manager_ptr,
221     message_hub_ptr
222 )
```

```

223 {
224     // 1. set attributes
225
226     // 1.1. private
227     //...
228
229     // 1.2. public
230     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
231
232     this->is_running = false;
233
234     this->health = 100;
235
236     this->capacity_kW = 100;
237
238     this->production_MWh = 0;
239     this->max_production_MWh = 72;
240
241     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
242     this->smoke_dx = 5 * SECONDS_PER_FRAME;
243     this->smoke_dy = -10 * SECONDS_PER_FRAME;
244     this->smoke_prob = 8 * SECONDS_PER_FRAME;
245
246     this->smoke_sprite_list = {};
247
248     this->tile_improvement_string = "DIESEL GEN";
249
250     this->__setUpTileImprovementSpriteAnimated();
251
252     std::cout << "DieselGenerator constructed at " << this << std::endl;
253
254     return;
255 } /* DieselGenerator() */

```

4.3.2.2 ~DieselGenerator()

```

DieselGenerator::~DieselGenerator (
    void ) [virtual]

```

Destructor for the [DieselGenerator](#) class.

```

424 {
425     std::cout << "DieselGenerator at " << this << " destroyed" << std::endl;
426
427     return;
428 } /* ~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](#).

```

114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118         default: {
119             // do nothing!
120
121             break;
122         }
123     }
124 }
125
126 return;
127 } /* __handleKeyPressEvents() */

```

4.3.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146             break;
147         }
148     }
149
150     case (sf::Mouse::Right): {
151         //...
152         break;
153     }
154
155     default: {
156         // do nothing!
157         break;
158     }
159 }
160
161 return;
162 } /* __handleMouseButtonEvents() */
```

4.3.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet (
70         *(this->assets_manager_ptr->getTexture("diesel generator"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("diesel generator")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 } /* __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](#).

```
368 {
369     // 1. if just built, call base method and return
370     if (this->just_built) {
371         TileImprovement :: draw();
372     }
373     return;
374 }
375
376 // 1. draw first element of animated sprite
377 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
378
379 // 2. draw second element of animated sprite
380 if (this->is_running) {
381     //...
382 }
383
384 else {
385     //...
386 }
387
388 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
389
390 // 3. draw smoke effects
391 if (this->is_running) {
392     //...
393 }
394
395 // 4. draw production menu
396 if (this->production_menu_open) {
397     this->render_window_ptr->draw(this->production_menu_backing);
398     this->render_window_ptr->draw(this->production_menu_backing_text);
399 }
400
401 //...
402 }
403
404 this->frame++;
405 return;
406 }
407
408 /* draw() */
```

4.3.3.5 getTileOptionsSubstring()

```
std::string DieselGenerator::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
272 {
273     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
274
275     // 32 char x 17 line console "-----\n";
276     std::string options_substring = "CAPACITY: ";
277     options_substring += std::to_string(this->capacity_kW);
278     options_substring += " kW\n";
```

```

279
280     options_substring      += "PRODUCTION: ";
281     options_substring      += std::to_string(this->production_MWh);
282     options_substring      += " MWh (MAX ";
283     options_substring      += std::to_string(this->max_production_MWh);
284     options_substring      += ")\n";
285
286     options_substring      += "HEALTH: ";
287     options_substring      += std::to_string(this->health);
288     options_substring      += "/100\n";
289
290     options_substring      += "
291     options_substring      += "    **** DIESEL GEN OPTIONS **** \n";
292     options_substring      += "
293     options_substring      += "[E]:  OPEN PRODUCTION MENU \n";
294
295     options_substring      += "[U]:  UPGRADE (";
296     options_substring      += std::to_string(upgrade_cost);
297     options_substring      += " K)\n";
298
299     options_substring      += "[P]:  SCRAP (";
300     options_substring      += std::to_string(SCRAP_COST);
301     options_substring      += " K)";
302
303     return options_substring;
304 } /* getTileOptionsSubstring() */

```

4.3.3.6 processEvent()

```

void DieselGenerator::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

319 {
320     TileImprovement :: processEvent ();
321
322     if (this->event_ptr->type == sf::Event::KeyPressed) {
323         this->__handleKeyPressEvents();
324     }
325
326     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
327         this->__handleMouseButtonEvents();
328     }
329
330     return;
331 } /* processEvent() */

```

4.3.3.7 processMessage()

```

void DieselGenerator::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

346 {
347     TileImprovement :: processMessage ();
348
349     //...
350
351     return;
352 } /* processMessage() */

```

4.3.4 Member Data Documentation

4.3.4.1 capacity_kW

```
int DieselGenerator::capacity_kW
```

The rated production capacity [kW] of the diesel generator.

4.3.4.2 max_production_MWh

```
int DieselGenerator::max_production_MWh
```

The maximum production [MWh] for this turn.

4.3.4.3 production_MWh

```
int DieselGenerator::production_MWh
```

The current production [MWh] of the diesel generator.

4.3.4.4 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.5 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.6 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.7 smoke_prob

```
double DieselGenerator::smoke_prob
```

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

4.3.4.8 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:

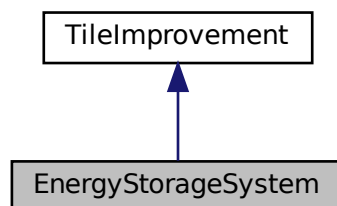
- header/[DieselGenerator.h](#)
- source/[DieselGenerator.cpp](#)

4.4 EnergyStorageSystem Class Reference

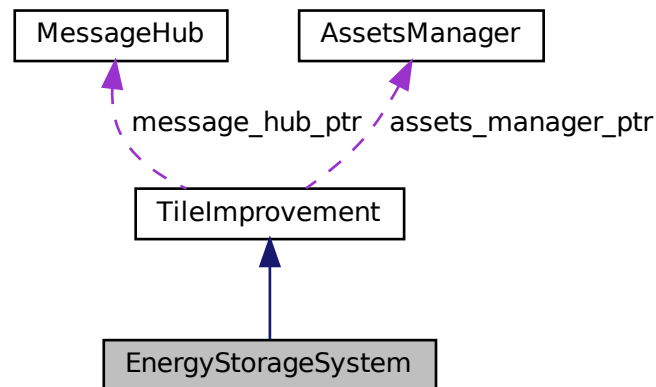
A settlement class (child class of [TileImprovement](#)).

```
#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.
- std::string [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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.

```
201 :
202 TileImprovement (
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
210 {
211     // 1. set attributes
212
213     // 1.1. private
214     //...
215
216     // 1.2. public
217     this->tile_improvement_type = TileImprovementType :: ENERGY_STORAGE_SYSTEM;
218
219     this->is_running = false;
220
221     this->health = 100;
222
223     this->tile_improvement_string = "ENERGY STORAGE";
224
225     this->__setUpTileImprovementSpriteStatic();
226
227     std::cout << "EnergyStorageSystem constructed at " << this << std::endl;
228
229     return;
230 } /* EnergyStorageSystem() */
```

4.4.2.2 ~EnergyStorageSystem()

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

Destructor for the [EnergyStorageSystem](#) class.

```

352 {
353     std::cout << "EnergyStorageSystem at " << this << " destroyed" << std::endl;
354
355     return;
356 } /* ~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](#).

```

103 {
104     switch (this->event_ptr->key.code) {
105         //...
106
107         default: {
108             // do nothing!
109
110             break;
111         }
112     }
113 }
114
115 return;
116 } /* __handleKeyPressEvents() */

```

4.4.3.2 __handleMouseButtonEvents()

```

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

```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```

131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139         case (sf::Mouse::Right): {
140             //...
141
142             break;
143         }
144
145         default: {
146             // do nothing!
147
148             break;
149         }
150     }
151 }
152
153 return;
154 } /* __handleMouseButtonEvents() */

```

4.4.3.3 __setUpTileImprovementSpriteStatic()

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

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

```
68 {
69     this->tile_improvement_sprite_static.setTexture(
70         *(this->assets_manager_ptr->getTexture("energy storage system"))
71     );
72     this->tile_improvement_sprite_static.setOrigin(
73         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
74         this->tile_improvement_sprite_static.getLocalBounds().height
75     );
76     this->tile_improvement_sprite_static.setPosition(
77         this->position_x,
78         this->position_y - 32
79     );
80     this->tile_improvement_sprite_static.setColor(
81         sf::Color(255, 255, 255, 0)
82     );
83     return;
84 } /* __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](#).

```
323 {
324     // 1. if just built, call base method and return
325     if (this->just_built) {
326         TileImprovement::draw();
327     }
328     return;
329 }
330
331 // 1. draw static sprite
332 this->render_window_ptr->draw(this->tile_improvement_sprite_static);
333
334 this->frame++;
335 return;
336 } /* draw() */
```

4.4.3.5 getTileOptionsSubstring()

```
std::string EnergyStorageSystem::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

247 {
248     //          32 char x 17 line console "-----\n";
249     std::string options_substring = "**** ENERGY STORAGE OPTIONS ****\n";
250     options_substring += "\n";
251     options_substring += "\n";
252     options_substring += "\n";
253     options_substring += "\n";
254     options_substring += "\n";
255     options_substring += "\n";
256     options_substring += "\n";
257
258     options_substring += "[P]: SCRAP ";
259     options_substring += std::to_string(SCRAP_COST);
260     options_substring += " K)";
261
262     return options_substring;
263 } /* getTileOptionsSubstring() */

```

4.4.3.6 processEvent()

```

void EnergyStorageSystem::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

278 {
279     if (this->event_ptr->type == sf::Event::KeyPressed) {
280         this->__handleKeyPressEvents();
281     }
282
283     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
284         this->__handleMouseButtonEvents();
285     }
286
287     return;
288 } /* processEvent() */

```

4.4.3.7 processMessage()

```

void EnergyStorageSystem::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

303 {
304     //...
305
306     return;
307 } /* processMessage() */

```

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

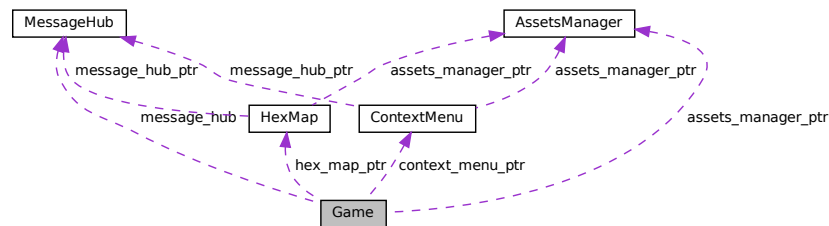
- header/[EnergyStorageSystem.h](#)
- source/[EnergyStorageSystem.cpp](#)

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.

```
702 {
703     // 1. set attributes
704
705     // 1.1. private
706     this->render_window_ptr = render_window_ptr;
707
708     this->assets_manager_ptr = assets_manager_ptr;
709
710     // 1.2. public
711     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
712
713     this->quit_game = false;
714     this->game_loop_broken = false;
715     this->show_frame_clock_overlay = false;
716
717     this->frame = 0;
718     this->time_since_start_s = 0;
719
720     double seconds_since_epoch = time(NULL);
721     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
722
723     this->year = 1970 + (int)years_since_epoch;
724     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
725
726     this->population = 0;
727     this->credits = STARTING_CREDITS;
728     this->demand_MWh = 0;
729     this->cumulative_emissions_tonnes = 0;
730
731     this->hex_map_ptr = new HexMap(
732         6,
733         &(this->event),
734         this->render_window_ptr,
735         this->assets_manager_ptr,
736         &(this->message_hub)
737     );
738
739     this->context_menu_ptr = new ContextMenu(
740         &(this->event),
741         this->render_window_ptr,
742         this->assets_manager_ptr,
743         &(this->message_hub)
744     );
745
746     // 2. add message channel(s)
747     this->message_hub.addChannel(GAME_CHANNEL);
748     this->message_hub.addChannel(GAME_STATE_CHANNEL);
749
750     std::cout << "Game constructed at " << this << std::endl;
751
752     return;
753 } /* Game() */
```

4.5.2.2 ~Game()

```
Game::~~Game (
    void )
```

Destructor for the [Game](#) class.

```
837 {
838     // 1. clean up attributes
839     delete this->hex_map_ptr;
840     delete this->context_menu_ptr;
841
842     std::cout << "Game at " << this << " destroyed" << std::endl;
843
844     return;
845 } /* ~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.

```
669 {
670     this->__drawHUD();
671
672     if (this->show_frame_clock_overlay) {
673         this->__drawFrameClockOverlay();
674     }
675
676     return;
677 } /* draw() */
```

4.5.3.2 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
495 {
496     std::string frame_clock_string = "FRAME: ";
497     frame_clock_string += std::to_string(this->frame);
498     frame_clock_string += "\nTIME SINCE START [s]: ";
499     frame_clock_string += std::to_string(this->time_since_start_s);
500
501     sf::Text frame_clock_text(
502         frame_clock_string,
503         *(this->assets_manager_ptr->getFont("DroidSansMono")),
504         16
505     );
506
507     sf::RectangleShape frame_clock_backing(
508         sf::Vector2f(
509             1.02 * frame_clock_text.getLocalBounds().width,
510             1.20 * frame_clock_text.getLocalBounds().height
511         )
512     );
513     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
514
515     this->render_window_ptr->draw(frame_clock_backing);
516     this->render_window_ptr->draw(frame_clock_text);
517
518     return;
519 } /* __drawFrameClockOverlay() */
```


4.5.3.3 __drawHUD()

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

Helper method to heads-up display (HUD).

```
534 {
535     // 1. first line (top)
536     std::string HUD_string = "YEAR: ";
537     HUD_string += std::to_string(this->year);
538
539     HUD_string += "    MONTH: ";
540     HUD_string += std::to_string(this->month);
541
542     HUD_string += "    POPULATION: ";
543     HUD_string += std::to_string(this->population);
544
545     HUD_string += "    CREDITS: ";
546     HUD_string += std::to_string(this->credits);
547     HUD_string += " K";
548
549     HUD_string += "    CURRENT DEMAND: ";
550     HUD_string += std::to_string(this->demand_MWh);
551     HUD_string += " MWh";
552
553     sf::Text HUD_text(
554         HUD_string,
555         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
556         16
557     );
558
559     HUD_text.setPosition(
560         (800 - HUD_text.getLocalBounds().width) / 2,
561         8
562     );
563
564     HUD_text.setFillColor(MONOCROME_TEXT_GREEN);
565
566     this->render_window_ptr->draw(HUD_text);
567
568
569     // 2. second line (top)
570     HUD_string = "CUMULATIVE EMISSIONS: ";
571     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
572     HUD_string += " tonnes (CO2e)";
573
574     HUD_string += "    LIFETIME LIMIT: ";
575     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
576     HUD_string += " tonnes (CO2e)";
577
578     HUD_text.setString(HUD_string);
579
580     HUD_text.setPosition(
581         (800 - HUD_text.getLocalBounds().width) / 2,
582         35
583     );
584
585     this->render_window_ptr->draw(HUD_text);
586
587
588     // 3. third line (bottom)
589     HUD_string = "GAME PHASE: ";
590
591     switch (this->game_phase) {
592     case (GamePhase :: BUILD_SETTLEMENT): {
593         HUD_string += "BUILD SETTLEMENT";
594
595         break;
596     }
597
598
599     case (GamePhase :: SYSTEM_MANAGEMENT): {
600         HUD_string += "SYSTEM MANAGEMENT";
601
602         break;
603     }
604
605
606     case (GamePhase :: LOSS_EMISSIONS): {
607         HUD_string += "LOSS (EMISSIONS)";
608
609         break;
610     }
611 }
```

```

612
613     case (GamePhase :: LOSS_DEMAND): {
614         HUD_string += "LOSS (DEMAND)";
615
616         break;
617     }
618
619
620     case (GamePhase :: LOSS_CREDITS): {
621         HUD_string += "LOSS (CREDITS)";
622
623         break;
624     }
625
626
627     case (GamePhase :: VICTORY): {
628         HUD_string += "VICTORY";
629
630         break;
631     }
632
633
634     default: {
635         HUD_string += "???";
636
637         break;
638     }
639 }
640
641 HUD_string += "    TURN: ";
642 HUD_string += std::to_string(this->turn);
643
644 HUD_text.setString(HUD_string);
645
646 HUD_text.setPosition(
647     (800 - HUD_text.getLocalBounds().width) / 2,
648     GAME_HEIGHT - 35
649 );
650
651 this->render_window_ptr->draw(HUD_text);
652
653 return;
654 } /* __drawHUD() */

```

4.5.3.4 __handleKeyPressEvents()

```

void Game::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

93 {
94     switch (this->event.key.code) {
95         case (sf::Keyboard::Tilde): {
96             this->__toggleFrameClockOverlay();
97
98             break;
99         }
100
101
102         case (sf::Keyboard::Tab): {
103             this->hex_map_ptr->toggleResourceOverlay();
104
105             break;
106         }
107
108
109         default: {
110             // do nothing!
111
112             break;
113         }
114     }
115
116     return;
117 } /* __handleKeyPressEvents() */

```

4.5.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
132 {
133     switch (this->event.mouseButton.button) {
134         case (sf::Mouse::Left): {
135             //...
136
137             break;
138         }
139
140
141         case (sf::Mouse::Right): {
142             //...
143
144             break;
145         }
146
147
148         default: {
149             // do nothing!
150
151             break;
152         }
153     }
154
155     return;
156 } /* __handleMouseButtonEvents() */
```

4.5.3.6 __insufficientCreditsAlarm()

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

Helper method to sound and display and insufficient credits alarm.

```
388 {
389     // 1. sound buzzer
390     this->assets_manager_ptr->getSound("insufficient credits")->play();
391
392     // 2. construct alarm text and backing rectangle
393     sf::Text insufficient_credits_text(
394         "INSUFFICIENT CREDITS",
395         (*this->assets_manager_ptr->getFont("DroidSansMono")),
396         32
397     );
398
399     insufficient_credits_text.setOrigin(
400         insufficient_credits_text.getLocalBounds().width / 2,
401         insufficient_credits_text.getLocalBounds().height / 2
402     );
403
404     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
405
406     sf::RectangleShape backing_rectangle(
407         sf::Vector2f(
408             1.1 * insufficient_credits_text.getLocalBounds().width,
409             1.5 * insufficient_credits_text.getLocalBounds().height
410         )
411     );
412
413     backing_rectangle.setFill-color(RESOURCE_CHIP_GREY);
414
415     backing_rectangle.setOrigin(
416         backing_rectangle.getLocalBounds().width / 2,
417         backing_rectangle.getLocalBounds().height / 2
418     );
419
420     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
421
422     // 3. display loop (blocking ~3 seconds)
423     bool red_flag = true;
424     int alarm_frame = 0;
```

```

425     double time_since_alarm_s = 0;
426
427     sf::Clock alarm_clock;
428
429     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
430
431         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
432
433         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
434             while (this->render_window_ptr->pollEvent(this->event)) {
435                 // do nothing!
436             }
437
438             this->render_window_ptr->clear();
439
440             this->hex_map_ptr->draw();
441             this->context_menu_ptr->draw();
442             this->__draw();
443
444             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
445                 if (red_flag) {
446                     red_flag = false;
447                 }
448
449                 else {
450                     red_flag = true;
451                 }
452             }
453
454             if (red_flag) {
455                 insufficient_credits_text.setFillColor(MONOCHROME_TEXT_RED);
456             }
457
458             else {
459                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
460             }
461
462             this->render_window_ptr->draw(backing_rectangle);
463             this->render_window_ptr->draw(insufficient_credits_text);
464
465             this->render_window_ptr->display();
466
467             alarm_frame++;
468             this->frame++;
469         }
470     }
471
472     // check track status, move to next if stopped
473     if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
474         this->assets_manager_ptr->nextTrack();
475         this->assets_manager_ptr->playTrack();
476     }
477 }
478
479 return;
480 } /* __insufficientCreditsAlarm( */

```

4.5.3.7 __processEvent()

```

void Game::__processEvent (
    void ) [private]

```

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

```

172 {
173     if (this->event.type == sf::Event::Closed) {
174         this->quit_game = true;
175         this->game_loop_broken = true;
176     }
177
178     if (this->event.type == sf::Event::KeyPressed) {
179         this->__handleKeyPressEvents();
180     }
181
182     if (this->event.type == sf::Event::MouseButtonPressed) {
183         this->__handleMouseButtonEvents();
184     }
185
186     return;
187 } /* __processEvent() */

```

4.5.3.8 __processMessage()

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

Helper method to process `Game`. To be called once per message.

```
285 {
286     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
287         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
288
289         if (game_channel_message.subject == "quit game") {
290             this->quit_game = true;
291             this->game_loop_broken = true;
292
293             std::cout << "Quit game message received by " << this << std::endl;
294             this->message_hub.popMessage(GAME_CHANNEL);
295         }
296
297         if (game_channel_message.subject == "restart game") {
298             this->game_loop_broken = true;
299
300             std::cout << "Restart game message received by " << this << std::endl;
301             this->message_hub.popMessage(GAME_CHANNEL);
302         }
303
304         if (game_channel_message.subject == "state request") {
305             std::cout << "Game state request message received by " << this << std::endl;
306
307             this->__sendGameStateMessage();
308             this->message_hub.popMessage(GAME_CHANNEL);
309         }
310
311         if (game_channel_message.subject == "credits spent") {
312             this->credits -= game_channel_message.int_payload["credits spent"];
313
314             std::cout << "Credits spent message ( " <<
315                 game_channel_message.int_payload["credits spent"] << " ) received by "
316                 << this << std::endl;
317
318             std::cout << "Current credits (Game): " << this->credits << " K" <<
319                 std::endl;
320
321             this->message_hub.popMessage(GAME_CHANNEL);
322         }
323
324         if (game_channel_message.subject == "insufficient credits") {
325             std::cout << "Insufficient credits message received by " << this <<
326                 std::endl;
327
328             this->__insufficientCreditsAlarm();
329
330             this->message_hub.popMessage(GAME_CHANNEL);
331         }
332
333         if (game_channel_message.subject == "update game phase") {
334             std::cout << "Update game phase message received by " << this << std::endl;
335
336             if (
337                 game_channel_message.string_payload["game phase"] == "system management"
338             ) {
339                 this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
340                 this->population = STARTING_POPULATION;
341                 this->turn++;
342             }
343
344             else if (
345                 game_channel_message.string_payload["game phase"] == "loss emissions"
346             ) {
347                 this->game_phase = GamePhase :: LOSS_EMISSIONS;
348             }
349
350             else if (
351                 game_channel_message.string_payload["game phase"] == "loss demand"
352             ) {
353                 this->game_phase = GamePhase :: LOSS_DEMAND;
354             }
355
356             else if (
357                 game_channel_message.string_payload["game phase"] == "loss credits"
358             ) {
359                 this->game_phase = GamePhase :: LOSS_CREDITS;
360             }
361
362             else if (
```

```

363         game_channel_message.string_payload["game phase"] == "victory"
364     ) {
365         this->game_phase = GamePhase :: VICTORY;
366     }
367
368     this->message_hub.popMessage(GAME_CHANNEL);
369 }
370 }
371
372 return;
373 } /* __processMessage() */

```

4.5.3.9 __sendGameStateMessage()

```

void Game::__sendGameStateMessage (
    void ) [private]

```

Helper method to format and send a game state message.

```

202 {
203     Message game_state_message;
204
205     game_state_message.channel = GAME_STATE_CHANNEL;
206     game_state_message.subject = "game state";
207
208     game_state_message.int_payload["year"] = this->year;
209     game_state_message.int_payload["month"] = this->month;
210     game_state_message.int_payload["population"] = this->population;
211     game_state_message.int_payload["credits"] = this->credits;
212     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
213     game_state_message.int_payload["cumulative_emissions_tonnes"] =
214         this->cumulative_emissions_tonnes;
215
216     switch (this->game_phase) {
217         case (GamePhase :: BUILD_SETTLEMENT): {
218             game_state_message.string_payload["game phase"] = "build settlement";
219             break;
220         }
221
222         case (GamePhase :: SYSTEM_MANAGEMENT): {
223             game_state_message.string_payload["game phase"] = "system management";
224             break;
225         }
226
227         case (GamePhase :: LOSS_EMISSIONS): {
228             game_state_message.string_payload["game phase"] = "loss emissions";
229             break;
230         }
231
232         case (GamePhase :: LOSS_DEMAND): {
233             game_state_message.string_payload["game phase"] = "loss demand";
234             break;
235         }
236
237         case (GamePhase :: LOSS_CREDITS): {
238             game_state_message.string_payload["game phase"] = "loss credits";
239             break;
240         }
241
242         case (GamePhase :: VICTORY): {
243             game_state_message.string_payload["game phase"] = "victory";
244             break;
245         }
246
247         default: {
248             // do nothing!
249         }
250     }
251 }

```

```

262         break;
263     }
264 }
265
266 this->message_hub.sendMessage(game_state_message);
267
268 std::cout << "Game state message sent by " << this << std::endl;
269 return;
270 } /* __sendGameStateMessage() */

```

4.5.3.10 __toggleFrameClockOverlay()

```

void Game::__toggleFrameClockOverlay (
    void ) [private]

```

Helper method to toggle frame clock overlay.

```

68 {
69     if (this->show_frame_clock_overlay) {
70         this->show_frame_clock_overlay = false;
71     }
72
73     else {
74         this->show_frame_clock_overlay = true;
75     }
76
77     return;
78 } /* __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).

```

771 {
772     // 1. play brand animation
773     //...
774
775     // 2. show splash screen
776     //...
777
778     // 3. start game loop
779     while (not this->game_loop_broken) {
780         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
781
782         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
783             // 6.1. process events
784             while (this->render_window_ptr->pollEvent(this->event)) {
785                 this->hex_map_ptr->processEvent();
786                 this->context_menu_ptr->processEvent();
787                 this->__processEvent();
788             }
789
790             // 6.2. process messages
791             while (this->message_hub.hasTraffic()) {
792                 this->hex_map_ptr->processMessage();
793                 this->context_menu_ptr->processMessage();
794                 this->__processMessage();
795             }
796         }
797     }
798 }

```

```

799         // 6.3. draw frame
800         this->render_window_ptr->clear();
801
802         this->hex_map_ptr->draw();
803         this->context_menu_ptr->draw();
804         this->__draw();
805
806         this->render_window_ptr->display();
807
808
809         // 6.4. increment frame
810         this->frame++;
811     }
812
813     // check track status, move to next if stopped
814     if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
815         this->assets_manager_ptr->nextTrack();
816         this->assets_manager_ptr->playTrack();
817     }
818
819 }
820
821 return this->quit_game;
822 } /* 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:

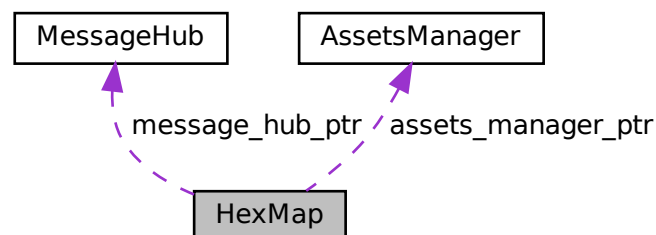
- header/[Game.h](#)
- source/[Game.cpp](#)

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.
- unsigned long long 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.

```
1116 {
1117     // 1. set attributes
1118
1119     // 1.1. private
1120     this->event_ptr = event_ptr;
1121     this->render_window_ptr = render_window_ptr;
1122
1123     this->assets_manager_ptr = assets_manager_ptr;
1124     this->message_hub_ptr = message_hub_ptr;
1125
1126     // 1.2. public
1127     this->show_resource = false;
1128     this->tile_selected = false;
1129
1130     this->frame = 0;
1131
1132     this->n_layers = n_layers;
1133     if (this->n_layers < 0) {
1134         this->n_layers = 0;
1135     }
1136
1137     this->position_x = 400;
1138     this->position_y = 400;
1139
1140     // 2. assemble n layer hex map
1141     this->__assembleHexMap();
1142
1143     // 3. set up and position drawable attributes
1144     this->__setUpGlassScreen();
1145
1146     // 4. add message channel(s)
1147     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1148     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1149     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1150     this->message_hub_ptr->addChannel(HEX_MAP_CHANNEL);
1151
1152     std::cout << "HexMap constructed at " << this << std::endl;
1153 }
```

```

1154     return;
1155 }    /* HexMap(), intended */

```

4.6.2.2 ~HexMap()

```

HexMap::~~HexMap (
    void )

```

Destructor for the [HexMap](#) class.

```

1449 {
1450     this->clear();
1451
1452     std::cout << "HexMap at " << this << " destroyed" << std::endl;
1453
1454     return;
1455 }    /* ~HexMap() */

```

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

```

void HexMap::__assembleHexMap (
    void ) [private]

```

Helper method to assemble the hex map.

```

875 {
876     // 1. seed RNG (using milliseconds since 1 Jan 1970)
877     unsigned long long int milliseconds_since_epoch =
878         std::chrono::duration_cast<std::chrono::milliseconds>(
879             std::chrono::system_clock::now().time_since_epoch()
880         ).count();
881     srand(milliseconds_since_epoch);
882
883     // 2. lay tiles
884     this->__layTiles();
885     this->__buildDrawOrderVector();
886
887     // 3. procedurally generate types
888     this->__procedurallyGenerateTileTypes();
889
890     // 4. procedurally generate resources
891     this->__procedurallyGenerateTileResources();
892
893     return;
894 }    /* __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.
----------------	--

```

1067 {
1068     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
1069
1070     for (size_t i = 0; i < neighbours_vec.size(); i++) {
1071         neighbours_vec[i]->assess();
1072     }
1073
1074     return;
1075 } /* __assessNeighbours() */

```

4.6.3.3 __buildDrawOrderVector()

```

void HexMap::__buildDrawOrderVector (
    void ) [private]

```

Helper method to build tile drawing order vector.

```

273 {
274     // 1. build temp list of tiles
275     std::list<HexTile*> temp_list;
276
277     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
278     std::map<double, HexTile*>::iterator hex_map_iter_y;
279     for (
280         hex_map_iter_x = this->hex_map.begin();
281         hex_map_iter_x != this->hex_map.end();
282         hex_map_iter_x++
283     ) {
284         for (
285             hex_map_iter_y = hex_map_iter_x->second.begin();
286             hex_map_iter_y != hex_map_iter_x->second.end();
287             hex_map_iter_y++
288         ) {
289             temp_list.push_back(hex_map_iter_y->second);
290         }
291     }
292
293     // 2. move elements from temp list to drawing order vector
294     double min_position_y = 0;
295     std::list<HexTile*>::iterator list_iter;
296
297     while (not temp_list.empty()) {
298         // 2.1. determine min y position
299         min_position_y = std::numeric_limits<double>::infinity();
300
301         for (
302             list_iter = temp_list.begin();
303             list_iter != temp_list.end();
304             list_iter++
305         ) {
306             if ((*list_iter)->position_y < min_position_y) {
307                 min_position_y = (*list_iter)->position_y;
308             }
309         }
310
311         // 2.2 move min y list elements to drawing order vec
312         list_iter = temp_list.begin();
313         while (list_iter != temp_list.end()) {
314             if ((*list_iter)->position_y == min_position_y) {
315                 this->hex_draw_order_vec.push_back((*list_iter));
316                 list_iter = temp_list.erase(list_iter);
317             }
318             else {
319                 list_iter++;
320             }
321         }
322     }
323 }
324
325 return;
326 } /* __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.

```
786 {
787     std::cout << "enforcing ocean continuity ..." << std::endl;
788
789     bool tile_changed = false;
790
791     // 1. scan tiles and enforce (where appropriate)
792     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
793     std::map<double, HexTile*>::iterator hex_map_iter_y;
794     HexTile* hex_ptr;
795     for (
796         hex_map_iter_x = this->hex_map.begin();
797         hex_map_iter_x != this->hex_map.end();
798         hex_map_iter_x++
799     ) {
800         for (
801             hex_map_iter_y = hex_map_iter_x->second.begin();
802             hex_map_iter_y != hex_map_iter_x->second.end();
803             hex_map_iter_y++
804         ) {
805             hex_ptr = hex_map_iter_y->second;
806
807             if (this->__isLakeTouchingOcean(hex_ptr)) {
808                 hex_ptr->setTileType(TileType :: OCEAN);
809                 tile_changed = true;
810             }
811         }
812     }
813
814     if (tile_changed) {
815         this->__enforceOceanContinuity();
816     }
817     else {
818         return;
819     }
820 } /* __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

<code>hex_ptr</code>	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.

```
642 {
643     // 1. init type count map
644     std::map<TileType, int> type_count_map;
645     type_count_map[hex_ptr->tile_type] = 1;
646
647     // 2. survey neighbours, count type instances
```

```

648     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
649
650     for (size_t i = 0; i < neighbours_vec.size(); i++) {
651         if (type_count_map.count(neighbours_vec[i]->tile_type) <= 0) {
652             type_count_map[neighbours_vec[i]->tile_type] = 1;
653         }
654         else {
655             type_count_map[neighbours_vec[i]->tile_type] += 1;
656         }
657     }
658
659     // 3. find majority tile type
660     int max_count = -1 * std::numeric_limits<int>::infinity();
661     TileType majority_tile_type = hex_ptr->tile_type;
662
663     std::map<TileType, int>::iterator map_iter;
664     for (
665         map_iter = type_count_map.begin();
666         map_iter != type_count_map.end();
667         map_iter++
668     ){
669         if (map_iter->second > max_count) {
670             max_count = map_iter->second;
671             majority_tile_type = map_iter->first;
672         }
673     }
674
675     // 4. detect ties
676     for (
677         map_iter = type_count_map.begin();
678         map_iter != type_count_map.end();
679         map_iter++
680     ){
681         if (
682             map_iter->second == max_count and
683             map_iter->first != majority_tile_type
684         ) {
685             majority_tile_type = hex_ptr->tile_type;
686             break;
687         }
688     }
689
690     return majority_tile_type;
691 } /* __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.

```

584 {
585     std::vector<HexTile*> neighbours_vec;
586
587     // 1. build potential neighbour positions
588     std::vector<double> potential_neighbour_x_vec(6, 0);
589     std::vector<double> potential_neighbour_y_vec(6, 0);
590
591     for (int i = 0; i < 6; i++) {
592         potential_neighbour_x_vec[i] = hex_ptr->position_x +
593             2 * hex_ptr->minor_radius * cos((60 * i) * (M_PI / 180));
594
595         potential_neighbour_y_vec[i] = hex_ptr->position_y +

```

```

596         2 * hex_ptr->minor_radius * sin((60 * i) * (M_PI / 180));
597     }
598
599     // 2. populate neighbours vector
600     std::vector<double> map_index_positions;
601     double potential_x = 0;
602     double potential_y = 0;
603
604     for (int i = 0; i < 6; i++) {
605         potential_x = potential_neighbour_x_vec[i];
606         potential_y = potential_neighbour_y_vec[i];
607
608         map_index_positions = this->__getValidMapIndexPositions(
609             potential_x,
610             potential_y
611         );
612
613         if (not (map_index_positions[0] == -1)) {
614             neighbours_vec.push_back(
615                 this->hex_map[map_index_positions[0]][map_index_positions[1]]
616             );
617         }
618     }
619
620     return neighbours_vec;
621 } /* __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].

```

349 {
350     // 1. generate random amplitude, wave number, direction, and phase vectors
351     std::vector<double> random_amplitude_vec(n_components, 0);
352     std::vector<double> random_wave_number_vec(n_components, 0);
353     std::vector<double> random_frequency_vec(n_components, 0);
354     std::vector<double> random_direction_vec(n_components, 0);
355     std::vector<double> random_phase_vec(n_components, 0);
356
357     for (int i = 0; i < n_components; i++) {
358         random_amplitude_vec[i] = 10 * ((double)rand() / RAND_MAX);
359
360         random_wave_number_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
361
362         random_frequency_vec[i] = ((double)rand() / RAND_MAX);
363
364         random_direction_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
365
366         random_phase_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
367     }
368
369     // 2. generate noise vec
370     double amp = 0;
371     double wave_no = 0;
372     double freq = 0;
373     double dir = 0;

```

```

374     double phase = 0;
375
376     double x = 0;
377     double y = 0;
378     double t = time(NULL);
379
380     double max_noise = -1 * std::numeric_limits<double>::infinity();
381     double min_noise = std::numeric_limits<double>::infinity();
382
383     double noise = 0;
384     std::vector<double> noise_vec(n_elements, 0);
385
386     for (int i = 0; i < n_elements; i++) {
387         x = this->tile_position_x_vec[i] - this->position_x;
388         y = this->tile_position_y_vec[i] - this->position_y;
389
390         for (int j = 0; j < n_components; j++) {
391             amp = random_amplitude_vec[j];
392             wave_no = random_wave_number_vec[j];
393             freq = random_frequency_vec[j];
394             dir = random_direction_vec[j];
395             phase = random_phase_vec[j];
396
397             noise += (amp / (j + 1)) * cos(
398                 wave_no * (j + 1) * (x * sin(dir) + y * cos(dir)) +
399                 2 * M_PI * (j + 1) * freq * t +
400                 phase
401             );
402         }
403
404         noise_vec[i] = noise;
405
406         if (noise > max_noise) {
407             max_noise = noise;
408         }
409
410         else if (noise < min_noise) {
411             min_noise = noise;
412         }
413
414         noise = 0;
415     }
416
417     // 3. normalize noise vec
418     for (int i = 0; i < n_elements; i++) {
419         noise_vec[i] = (noise_vec[i] - min_noise) / (max_noise - min_noise);
420
421         if (noise_vec[i] < 0) {
422             noise_vec[i] = 0;
423         }
424         else if (noise_vec[i] > 1) {
425             noise_vec[i] = 1;
426         }
427     }
428
429     return noise_vec;
430 } /* __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).

```

911 {
912     HexTile* selected_tile_ptr = NULL;
913
914     bool break_flag = false;
915     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
916     std::map<double, HexTile*>::iterator hex_map_iter_y;
917

```

```

918     for (
919         hex_map_iter_x = this->hex_map.begin();
920         hex_map_iter_x != this->hex_map.end();
921         hex_map_iter_x++
922     ) {
923         for (
924             hex_map_iter_y = hex_map_iter_x->second.begin();
925             hex_map_iter_y != hex_map_iter_x->second.end();
926             hex_map_iter_y++
927         ) {
928             if (hex_map_iter_y->second->is_selected) {
929                 selected_tile_ptr = hex_map_iter_y->second;
930                 break_flag = true;
931             }
932
933             if (break_flag) {
934                 break;
935             }
936         }
937
938         if (break_flag) {
939             break;
940         }
941     }
942
943     return selected_tile_ptr;
944 } /* __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_x</i>	The potential x position of the tile.
<i>potential_y</i>	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.

```

530 {
531     std::vector<double> map_index_positions = {-1, -1};
532
533     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
534     std::map<double, HexTile*>::iterator hex_map_iter_y;
535     HexTile* hex_ptr;
536
537     double distance = 0;
538
539     for (
540         hex_map_iter_x = this->hex_map.begin();
541         hex_map_iter_x != this->hex_map.end();
542         hex_map_iter_x++
543     ) {
544         for (
545             hex_map_iter_y = hex_map_iter_x->second.begin();
546             hex_map_iter_y != hex_map_iter_x->second.end();
547             hex_map_iter_y++
548         ) {
549             hex_ptr = hex_map_iter_y->second;
550
551             distance = sqrt(

```

```

552         pow(hex_ptr->position_x - potential_x, 2) +
553         pow(hex_ptr->position_y - potential_y, 2)
554     );
555
556     if (distance <= hex_ptr->minor_radius / 4) {
557         map_index_positions = {hex_ptr->position_x, hex_ptr->position_y};
558         return map_index_positions;
559     }
560 }
561 }
562
563 return map_index_positions;
564 } /* __isInHexMap() */

```

4.6.3.10 __handleKeyPressEvents()

```

void HexMap::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

959 {
960     switch (this->event_ptr->key.code) {
961         case (sf::Keyboard::Escape): {
962             this->tile_selected = false;
963         }
964
965
966         default: {
967             // do nothing!
968
969             break;
970         }
971     }
972
973     return;
974 } /* __handleKeyPressEvents() */

```

4.6.3.11 __handleMouseButtonEvents()

```

void HexMap::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

989 {
990     switch (this->event_ptr->mouseButton.button) {
991         case (sf::Mouse::Left): {
992             HexTile* hex_ptr = this->__getSelectedTile();
993
994             if (hex_ptr != NULL) {
995                 this->tile_selected = true;
996             }
997
998             else if (this->tile_selected) {
999                 this->tile_selected = false;
1000                 this->__sendNoTileSelectedMessage();
1001             }
1002
1003             break;
1004         }
1005
1006
1007         case (sf::Mouse::Right): {
1008             if (this->tile_selected) {
1009                 this->tile_selected = false;
1010                 this->__sendNoTileSelectedMessage();
1011             }
1012
1013             break;
1014         }

```

```

1015
1016
1017         default: {
1018             // do nothing!
1019
1020             break;
1021         }
1022     }
1023
1024     return;
1025 } /* __handleMouseButtonEvents() */

```

4.6.3.12 __isLakeTouchingOcean()

```

bool HexMap::__isLakeTouchingOcean (
    HexTile * hex_ptr ) [private]
753 {
754     // 1. if not lake tile, return
755     if (not (hex_ptr->tile_type == TileType :: LAKE)) {
756         return false;
757     }
758
759     // 2. scan neighbours for ocean tiles
760     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
761
762     for (size_t i = 0; i < neighbours_vec.size(); i++) {
763         if (neighbours_vec[i]->tile_type == TileType :: OCEAN) {
764             return true;
765         }
766     }
767
768     return false;
769 } /* __isLakeTouchingOcean() */

```

4.6.3.13 __layTiles()

```

void HexMap::__layTiles (
    void ) [private]

```

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

```

88 {
89     this->n_tiles = 0;
90
91     // 1. add origin tile
92     HexTile* hex_ptr = new HexTile(
93         this->position_x,
94         this->position_y,
95         this->event_ptr,
96         this->render_window_ptr,
97         this->assets_manager_ptr,
98         this->message_hub_ptr
99     );
100
101     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
102     this->tile_position_x_vec.push_back(hex_ptr->position_x);
103     this->tile_position_y_vec.push_back(hex_ptr->position_y);
104     this->n_tiles++;
105
106
107     // 2. fill out first row (reflect across origin tile)
108     for (int i = 0; i < this->n_layers; i++) {
109         hex_ptr = new HexTile(
110             this->position_x + 2 * (i + 1) * hex_ptr->minor_radius,
111             this->position_y,
112             this->event_ptr,
113             this->render_window_ptr,
114             this->assets_manager_ptr,
115             this->message_hub_ptr
116         );
117

```

```

118     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
119     this->tile_position_x_vec.push_back(hex_ptr->position_x);
120     this->tile_position_y_vec.push_back(hex_ptr->position_y);
121     this->n_tiles++;
122
123     if (i == this->n_layers - 1) {
124         this->border_tiles_vec.push_back(hex_ptr);
125     }
126
127     hex_ptr = new HexTile(
128         this->position_x - 2 * (i + 1) * hex_ptr->minor_radius,
129         this->position_y,
130         this->event_ptr,
131         this->render_window_ptr,
132         this->assets_manager_ptr,
133         this->message_hub_ptr
134     );
135
136     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
137     this->tile_position_x_vec.push_back(hex_ptr->position_x);
138     this->tile_position_y_vec.push_back(hex_ptr->position_y);
139     this->n_tiles++;
140
141     if (i == this->n_layers - 1) {
142         this->border_tiles_vec.push_back(hex_ptr);
143     }
144 }
145
146 // 3. fill out subsequent rows (reflect across first row)
147 HexTile* first_row_left_tile = hex_ptr;
148
149 int offset_count = 1;
150
151 double x_offset = 0;
152 double y_offset = 0;
153
154 for (
155     int row_width = 2 * this->n_layers;
156     row_width > this->n_layers;
157     row_width--
158 ) {
159     // 3.1. upper row
160     x_offset = first_row_left_tile->position_x +
161         2 * offset_count * first_row_left_tile->minor_radius *
162         cos(60 * (M_PI / 180));
163
164     y_offset = first_row_left_tile->position_y -
165         2 * offset_count * first_row_left_tile->minor_radius *
166         sin(60 * (M_PI / 180));
167
168     hex_ptr = new HexTile(
169         x_offset,
170         y_offset,
171         this->event_ptr,
172         this->render_window_ptr,
173         this->assets_manager_ptr,
174         this->message_hub_ptr
175     );
176
177     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
178     this->tile_position_x_vec.push_back(hex_ptr->position_x);
179     this->tile_position_y_vec.push_back(hex_ptr->position_y);
180     this->n_tiles++;
181
182     this->border_tiles_vec.push_back(hex_ptr);
183
184     for (int i = 1; i < row_width; i++) {
185         x_offset += 2 * first_row_left_tile->minor_radius;
186
187         hex_ptr = new HexTile(
188             x_offset,
189             y_offset,
190             this->event_ptr,
191             this->render_window_ptr,
192             this->assets_manager_ptr,
193             this->message_hub_ptr
194         );
195
196         this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
197         this->tile_position_x_vec.push_back(hex_ptr->position_x);
198         this->tile_position_y_vec.push_back(hex_ptr->position_y);
199         this->n_tiles++;
200
201         if (row_width == this->n_layers + 1 or i == row_width - 1) {
202             this->border_tiles_vec.push_back(hex_ptr);
203         }
204     }

```



```

205     }
206
207     // 3.2. lower row
208     x_offset = first_row_left_tile->position_x +
209         2 * offset_count * first_row_left_tile->minor_radius *
210         cos(60 * (M_PI / 180));
211
212     y_offset = first_row_left_tile->position_y +
213         2 * offset_count * first_row_left_tile->minor_radius *
214         sin(60 * (M_PI / 180));
215
216     hex_ptr = new HexTile(
217         x_offset,
218         y_offset,
219         this->event_ptr,
220         this->render_window_ptr,
221         this->assets_manager_ptr,
222         this->message_hub_ptr
223     );
224
225     this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
226     this->tile_position_x_vec.push_back(hex_ptr->position_x);
227     this->tile_position_y_vec.push_back(hex_ptr->position_y);
228     this->n_tiles++;
229
230     this->border_tiles_vec.push_back(hex_ptr);
231
232     for (int i = 1; i < row_width; i++) {
233         x_offset += 2 * first_row_left_tile->minor_radius;
234
235         hex_ptr = new HexTile(
236             x_offset,
237             y_offset,
238             this->event_ptr,
239             this->render_window_ptr,
240             this->assets_manager_ptr,
241             this->message_hub_ptr
242         );
243
244         this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
245         this->tile_position_x_vec.push_back(hex_ptr->position_x);
246         this->tile_position_y_vec.push_back(hex_ptr->position_y);
247         this->n_tiles++;
248
249         if (row_width == this->n_layers + 1 or i == row_width - 1) {
250             this->border_tiles_vec.push_back(hex_ptr);
251         }
252     }
253
254     offset_count++;
255 }
256
257 return;
258 } /* __layTiles() */

```

4.6.3.14 __procedurallyGenerateTileResources()

```

void HexMap::__procedurallyGenerateTileResources (
    void ) [private]

```

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

```

835 {
836     // 1. get random cosine series noise vec
837     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
838
839     // 2. set tile resources based on random cosine series noise
840     int noise_idx = 0;
841
842     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
843     std::map<double, HexTile*>::iterator hex_map_iter_y;
844     for (
845         hex_map_iter_x = this->hex_map.begin();
846         hex_map_iter_x != this->hex_map.end();
847         hex_map_iter_x++
848     ) {
849         for (
850             hex_map_iter_y = hex_map_iter_x->second.begin();
851             hex_map_iter_y != hex_map_iter_x->second.end();

```

```

852         hex_map_iter_y++
853     ) {
854         hex_map_iter_y->second->setTileResource(noise_vec[noise_idx]);
855         noise_idx++;
856     }
857 }
858
859 return;
860 } /* __procedurallyGenerateTileResources() */

```

4.6.3.15 __procedurallyGenerateTileTypes()

```

void HexMap::__procedurallyGenerateTileTypes (
    void ) [private]

```

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

```

445 {
446     // 1. get random cosine series noise vec
447     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
448
449     // 2. set initial tile types based on either random cosine series noise or white
450     //     noise (decided by coin toss)
451     int noise_idx = 0;
452
453     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
454     std::map<double, HexTile*>::iterator hex_map_iter_y;
455     for (
456         hex_map_iter_x = this->hex_map.begin();
457         hex_map_iter_x != this->hex_map.end();
458         hex_map_iter_x++
459     ) {
460         for (
461             hex_map_iter_y = hex_map_iter_x->second.begin();
462             hex_map_iter_y != hex_map_iter_x->second.end();
463             hex_map_iter_y++
464         ) {
465             if ((double)rand() / RAND_MAX > 0.5) {
466                 hex_map_iter_y->second->setTileType(noise_vec[noise_idx]);
467             }
468             else {
469                 hex_map_iter_y->second->setTileType((double)rand() / RAND_MAX);
470             }
471             noise_idx++;
472         }
473     }
474
475     // 3. smooth tile types (majority rules)
476     this->__smoothTileTypes();
477
478     // 4. set border tile type to ocean
479     for (size_t i = 0; i < this->border_tiles_vec.size(); i++) {
480         this->border_tiles_vec[i]->setTileType(TileType :: OCEAN);
481     }
482
483     // 5. enforce ocean continuity (i.e. all lake tiles touching ocean become ocean)
484     this->__enforceOceanContinuity();
485
486     // 6. decorate tiles
487     for (
488         hex_map_iter_x = this->hex_map.begin();
489         hex_map_iter_x != this->hex_map.end();
490         hex_map_iter_x++
491     ) {
492         for (
493             hex_map_iter_y = hex_map_iter_x->second.begin();
494             hex_map_iter_y != hex_map_iter_x->second.end();
495             hex_map_iter_y++
496         ) {
497             hex_map_iter_y->second->decorateTile();
498         }
499     }
500
501     return;
502 } /* __procedurallyGenerateTileTypes() */

```

4.6.3.16 __sendNoTileSelectedMessage()

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

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

```
1040 {
1041     Message no_tile_selected_message;
1042
1043     no_tile_selected_message.channel = NO_TILE_SELECTED_CHANNEL;
1044     no_tile_selected_message.subject = "no tile selected";
1045
1046     this->message_hub_ptr->sendMessage(no_tile_selected_message);
1047
1048     std::cout << "No tile selected message sent by " << this << std::endl;
1049     return;
1050 } /* __sendNoTileSelectedMessage() */
```

4.6.3.17 __setUpGlassScreen()

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

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

```
68 {
69     this->glass_screen.setSize(sf::Vector2f(GAME_WIDTH, GAME_HEIGHT));
70     this->glass_screen.setFillColor(sf::Color(MONOCROME_SCREEN_BACKGROUND));
71
72     return;
73 } /* __setUpGlassScreen() */
```

4.6.3.18 __smoothTileTypes()

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

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

```
706 {
707     std::cout << "smoothing ..." << std::endl;
708
709     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
710     std::map<double, HexTile*>::iterator hex_map_iter_y;
711     HexTile* hex_ptr;
712     TileType majority_tile_type;
713
714     for (
715         hex_map_iter_x = this->hex_map.begin();
716         hex_map_iter_x != this->hex_map.end();
717         hex_map_iter_x++
718     ) {
719         for (
720             hex_map_iter_y = hex_map_iter_x->second.begin();
721             hex_map_iter_y != hex_map_iter_x->second.end();
722             hex_map_iter_y++
723         ) {
724             hex_ptr = hex_map_iter_y->second;
725             majority_tile_type = this->__getMajorityTileType(hex_ptr);
726
727             if (majority_tile_type != hex_ptr->tile_type) {
728                 hex_ptr->setTileType(majority_tile_type);
729             }
730         }
731     }
732
733     return;
734 } /* __smoothTileTypes() */
```

4.6.3.19 assess()

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

Method to assess the resource of the selected tile.

```
1170 {
1171     HexTile* selected_tile_ptr = this->__getSelectedTile();
1172     if (selected_tile_ptr != NULL) {
1173         selected_tile_ptr->assess();
1174     }
1175
1176     return;
1177 } /* assess() */
```

4.6.3.20 clear()

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

Method to clear the hex map.

```
1411 {
1412     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1413     std::map<double, HexTile*>::iterator hex_map_iter_y;
1414     for (
1415         hex_map_iter_x = this->hex_map.begin();
1416         hex_map_iter_x != this->hex_map.end();
1417         hex_map_iter_x++
1418     ) {
1419         for (
1420             hex_map_iter_y = hex_map_iter_x->second.begin();
1421             hex_map_iter_y != hex_map_iter_x->second.end();
1422             hex_map_iter_y++
1423         ) {
1424             delete hex_map_iter_y->second;
1425         }
1426     }
1427     this->hex_map.clear();
1428
1429     this->tile_position_x_vec.clear();
1430     this->tile_position_y_vec.clear();
1431     this->border_tiles_vec.clear();
1432
1433     return;
1434 } /* 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.

```
1348 {
1349     // 1. draw background
1350     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1351     glass_screen_colour.a = 255;
1352     this->glass_screen.setFillColor(glass_screen_colour);
1353
1354     this->render_window_ptr->draw(this->glass_screen);
1355
1356     // 2. draw tiles (other than the selected tile) in drawing order
1357     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1358         if (not this->hex_draw_order_vec[i]->is_selected) {
1359             this->hex_draw_order_vec[i]->draw();
1360         }
1361     }
```

```

1362
1363 // 3. draw selected tile
1364 HexTile* selected_tile_ptr = this->__getSelectedTile();
1365 if (selected_tile_ptr != NULL) {
1366     selected_tile_ptr->draw();
1367 }
1368
1369 // 4. draw resource overlay text indication
1370 if (this->show_resource) {
1371     sf::Text resource_overlay_text(
1372         "**** RENEWABLE RESOURCE OVERLAY ****",
1373         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1374         16
1375     );
1376
1377     resource_overlay_text.setPosition(
1378         (800 - resource_overlay_text.getLocalBounds().width) / 2,
1379         GAME_HEIGHT - 70
1380     );
1381
1382     resource_overlay_text.setFillColor(MONOCHROME_TEXT_GREEN);
1383
1384     this->render_window_ptr->draw(resource_overlay_text);
1385 }
1386
1387 // 5. draw glass screen
1388 glass_screen_colour = this->glass_screen.getFillColor();
1389 glass_screen_colour.a = 40;
1390 this->glass_screen.setFillColor(glass_screen_colour);
1391
1392 this->render_window_ptr->draw(this->glass_screen);
1393
1394 this->frame++;
1395 return;
1396 } /* draw() */

```

4.6.3.22 processEvent()

```

void HexMap::processEvent (
    void )

```

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

```

1255 {
1256     // 1. process HexTile events
1257     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1258     std::map<double, HexTile*>::iterator hex_map_iter_y;
1259     for (
1260         hex_map_iter_x = this->hex_map.begin();
1261         hex_map_iter_x != this->hex_map.end();
1262         hex_map_iter_x++
1263     ) {
1264         for (
1265             hex_map_iter_y = hex_map_iter_x->second.begin();
1266             hex_map_iter_y != hex_map_iter_x->second.end();
1267             hex_map_iter_y++
1268         ) {
1269             hex_map_iter_y->second->processEvent();
1270         }
1271     }
1272
1273     // 2. process HexMap events
1274     if (this->event_ptr->type == sf::Event::KeyPressed) {
1275         this->__handleKeyPressEvents();
1276     }
1277
1278     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1279         this->__handleMouseButtonEvents();
1280     }
1281
1282     return;
1283 } /* processEvent() */

```

4.6.3.23 processMessage()

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

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

```
1298 {
1299     // 1. process HexTile messages
1300     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1301     std::map<double, HexTile*>::iterator hex_map_iter_y;
1302     for (
1303         hex_map_iter_x = this->hex_map.begin();
1304         hex_map_iter_x != this->hex_map.end();
1305         hex_map_iter_x++
1306     ) {
1307         for (
1308             hex_map_iter_y = hex_map_iter_x->second.begin();
1309             hex_map_iter_y != hex_map_iter_x->second.end();
1310             hex_map_iter_y++
1311         ) {
1312             hex_map_iter_y->second->processMessage();
1313         }
1314     }
1315
1316     // 2. process HexMap messages
1317     if (not this->message_hub_ptr->isEmpty(HEX_MAP_CHANNEL)) {
1318         Message hex_map_message = this->message_hub_ptr->receiveMessage(
1319             HEX_MAP_CHANNEL
1320         );
1321
1322         if (hex_map_message.subject == "assess neighbours") {
1323             HexTile* hex_ptr = this->__getSelectedTile();
1324             this->__assessNeighbours(hex_ptr);
1325
1326             std::cout << "Assess neighbours message received by " << this << std::endl;
1327             this->message_hub_ptr->popMessage(HEX_MAP_CHANNEL);
1328         }
1329     }
1330
1331     return;
1332 } /* processMessage() */
```

4.6.3.24 reroll()

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

Method to re-roll the hex map.

```
1192 {
1193     this->clear();
1194     this->__assembleHexMap();
1195
1196     return;
1197 } /* reroll() */
```

4.6.3.25 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
1212 {
1213     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1214     std::map<double, HexTile*>::iterator hex_map_iter_y;
1215     for (
1216         hex_map_iter_x = this->hex_map.begin();
```

```

1217         hex_map_iter_x != this->hex_map.end();
1218         hex_map_iter_x++
1219     ) {
1220         for (
1221             hex_map_iter_y = hex_map_iter_x->second.begin();
1222             hex_map_iter_y != hex_map_iter_x->second.end();
1223             hex_map_iter_y++
1224         ) {
1225             hex_map_iter_y->second->toggleResourceOverlay();
1226         }
1227     }
1228
1229     if (this->show_resource) {
1230         this->show_resource = false;
1231         this->assets_manager_ptr->getSound("resource overlay toggle off")->play();
1232     }
1233
1234     else {
1235         this->show_resource = true;
1236         this->assets_manager_ptr->getSound("resource overlay toggle on")->play();
1237     }
1238
1239     return;
1240 } /* 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

`unsigned long long 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:

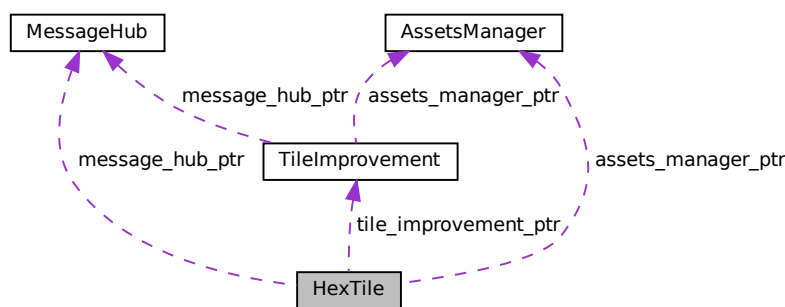
- header/[HexMap.h](#)
- source/[HexMap.cpp](#)

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.
- unsigned long long 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 [__scrapImprovement](#) (void)
Helper method to scrap the tile improvement ([Settlement](#) cannot be scrapped).
- 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.

```

2224 {
2225     // 1. set attributes
2226
2227     // 1.1. private
2228     this->event_ptr = event_ptr;
2229     this->render_window_ptr = render_window_ptr;
2230
2231     this->assets_manager_ptr = assets_manager_ptr;
2232     this->message_hub_ptr = message_hub_ptr;
2233
2234     // 1.2. public
2235     this->show_node = false;
2236     this->show_resource = false;
2237     this->resource_assessed = false;
2238     this->resource_assessment = false;
2239     this->is_selected = false;
2240     this->draw_explosion = false;
2241
2242     this->decoration_cleared = false;
2243     this->has_improvement = false;
2244     this->tile_improvement_ptr = NULL;
2245
2246     this->build_menu_open = false;
2247
2248     this->explosion_frame = 0;
2249
2250     this->frame = 0;
2251     this->credits = 0;
2252
2253     this->position_x = position_x;
2254     this->position_y = position_y;
2255
2256     this->major_radius = 32;
2257     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2258
2259     this->game_phase = "build settlement";
2260
2261     // 2. set up and position drawable attributes
2262     this->__setUpNodeSprite();
2263     this->__setUpTileSprite();
2264     this->__setUpSelectOutlineSprite();
2265     this->__setUpResourceChipSprite();
2266     this->__setUpResourceText();
2267     this->__setUpMagnifyingGlassSprite();
2268     this->__setUpTileExplosionReel();
2269
2270     // 3. set tile type and resource (default to none type and average)
2271     this->setTileType(TileType :: NONE_TYPE);
2272     this->setTileResource(TileResource :: AVERAGE);
2273
2274     std::cout << "HexTile constructed at " << this << std::endl;
2275
2276     return;
2277 } /* HexTile() */

```

4.7.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

2809 {
2810     if (this->tile_improvement_ptr != NULL) {

```

```

2811         delete this->tile_improvement_ptr;
2812     }
2813
2814     std::cout << "HexTile at " << this << " destroyed" << std::endl;
2815
2816     return;
2817 } /* ~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.

```

1370 {
1371     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1372
1373     if (this->credits < build_cost) {
1374         std::cout << "Cannot build diesel generator: insufficient credits (need "
1375             << build_cost << " K)" << std::endl;
1376
1377         this->__sendInsufficientCreditsMessage();
1378         return;
1379     }
1380
1381     this->tile_improvement_ptr = new DieselGenerator(
1382         this->position_x,
1383         this->position_y,
1384         this->event_ptr,
1385         this->render_window_ptr,
1386         this->assets_manager_ptr,
1387         this->message_hub_ptr
1388     );
1389
1390     this->has_improvement = true;
1391     this->__closeBuildMenu();
1392
1393     this->__sendCreditsSpentMessage(build_cost);
1394     this->__sendTileStateMessage();
1395     this->__sendGameStateRequest();
1396
1397     return;
1398 } /* __buildDieselGenerator() */

```

4.7.3.2 __buildEnergyStorage()

```

void HexTile::__buildEnergyStorage (
    void ) [private]

```

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

```

1613 {
1614     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1615
1616     if (this->credits < build_cost) {
1617         std::cout << "Cannot build energy storage system: insufficient credits (need "
1618             << build_cost << " K)" << std::endl;
1619
1620         this->__sendInsufficientCreditsMessage();
1621         return;
1622     }
1623
1624     this->tile_improvement_ptr = new EnergyStorageSystem(
1625         this->position_x,

```



```

1626         this->position_y,
1627         this->event_ptr,
1628         this->render_window_ptr,
1629         this->assets_manager_ptr,
1630         this->message_hub_ptr
1631     );
1632
1633     this->has_improvement = true;
1634     this->__closeBuildMenu();
1635
1636     this->__sendCreditsSpentMessage(build_cost);
1637     this->__sendTileStateMessage();
1638     this->__sendGameStateRequest();
1639
1640     return;
1641 } /* __buildEnergyStorage() */

```

4.7.3.3 __buildSettlement()

```

void HexTile::__buildSettlement (
    void ) [private]

```

Helper method to build a settlement on this tile.

```

1324 {
1325     if (this->credits < BUILD_SETTLEMENT_COST) {
1326         std::cout << "Cannot build settlement: insufficient credits (need "
1327             << BUILD_SETTLEMENT_COST << " K)" << std::endl;
1328
1329         this->__sendInsufficientCreditsMessage();
1330         return;
1331     }
1332
1333     this->__clearDecoration();
1334
1335     this->tile_improvement_ptr = new Settlement(
1336         this->position_x,
1337         this->position_y,
1338         this->event_ptr,
1339         this->render_window_ptr,
1340         this->assets_manager_ptr,
1341         this->message_hub_ptr
1342     );
1343
1344     this->has_improvement = true;
1345
1346     this->assess();
1347     this->__sendAssessNeighboursMessage();
1348
1349     this->__sendUpdateGamePhaseMessage("system management");
1350     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1351     this->__sendTileStateMessage();
1352     this->__sendGameStateRequest();
1353
1354     return;
1355 } /* __buildSettlement() */

```

4.7.3.4 __buildSolarPV()

```

void HexTile::__buildSolarPV (
    void ) [private]

```

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

```

1413 {
1414     int build_cost = SOLAR_PV_BUILD_COST;
1415
1416     if (this->tile_type == TileType::LAKE) {
1417         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1418     }
1419

```

```

1420     if (this->credits < build_cost) {
1421         std::cout << "Cannot build solar PV array: insufficient credits (need "
1422             << build_cost << " K)" << std::endl;
1423
1424         this->__sendInsufficientCreditsMessage();
1425         return;
1426     }
1427
1428     this->tile_improvement_ptr = new SolarPV(
1429         this->position_x,
1430         this->position_y,
1431         this->event_ptr,
1432         this->render_window_ptr,
1433         this->assets_manager_ptr,
1434         this->message_hub_ptr
1435     );
1436
1437     this->has_improvement = true;
1438     this->__closeBuildMenu();
1439
1440     if (this->tile_type == TileType::LAKE) {
1441         this->decoration_cleared = true;
1442         this->assets_manager_ptr->getSound("splash")->play();
1443     }
1444
1445     this->__sendCreditsSpentMessage(build_cost);
1446     this->__sendTileStateMessage();
1447     this->__sendGameStateRequest();
1448
1449     return;
1450 } /* __buildSolarPV() */

```

4.7.3.5 __buildTidalTurbine()

```

void HexTile::__buildTidalTurbine (
    void ) [private]

```

Helper method to build a tidal turbine on this tile.

```

1523 {
1524     int build_cost = TIDAL_TURBINE_BUILD_COST;
1525
1526     if (this->credits < build_cost) {
1527         std::cout << "Cannot build tidal turbine: insufficient credits (need "
1528             << build_cost << " K)" << std::endl;
1529
1530         this->__sendInsufficientCreditsMessage();
1531         return;
1532     }
1533
1534     this->tile_improvement_ptr = new TidalTurbine(
1535         this->position_x,
1536         this->position_y,
1537         this->event_ptr,
1538         this->render_window_ptr,
1539         this->assets_manager_ptr,
1540         this->message_hub_ptr
1541     );
1542
1543     this->has_improvement = true;
1544     this->decoration_cleared = true;
1545     this->assets_manager_ptr->getSound("splash")->play();
1546     this->__closeBuildMenu();
1547
1548     this->__sendCreditsSpentMessage(build_cost);
1549     this->__sendTileStateMessage();
1550     this->__sendGameStateRequest();
1551
1552     return;
1553 } /* __buildTidalTurbine() */

```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1568 {
1569     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1570
1571     if (this->credits < build_cost) {
1572         std::cout << "Cannot build wave energy converter: insufficient credits (need "
1573             << build_cost << " K)" << std::endl;
1574
1575         this->__sendInsufficientCreditsMessage();
1576         return;
1577     }
1578
1579     this->tile_improvement_ptr = new WaveEnergyConverter(
1580         this->position_x,
1581         this->position_y,
1582         this->event_ptr,
1583         this->render_window_ptr,
1584         this->assets_manager_ptr,
1585         this->message_hub_ptr
1586     );
1587
1588     this->has_improvement = true;
1589     this->decoration_cleared = true;
1590     this->assets_manager_ptr->getSound("splash")->play();
1591     this->__closeBuildMenu();
1592
1593     this->__sendCreditsSpentMessage(build_cost);
1594     this->__sendTileStateMessage();
1595     this->__sendGameStateRequest();
1596
1597     return;
1598 } /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1465 {
1466     int build_cost = WIND_TURBINE_BUILD_COST;
1467
1468     if (
1469         (this->tile_type == TileType :: LAKE) or
1470         (this->tile_type == TileType :: OCEAN)
1471     ) {
1472         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1473     }
1474
1475     if (this->credits < build_cost) {
1476         std::cout << "Cannot build wind turbine: insufficient credits (need "
1477             << build_cost << " K)" << std::endl;
1478
1479         this->__sendInsufficientCreditsMessage();
1480         return;
1481     }
1482
1483     this->tile_improvement_ptr = new WindTurbine(
1484         this->position_x,
1485         this->position_y,
1486         this->event_ptr,
1487         this->render_window_ptr,
1488         this->assets_manager_ptr,
1489         this->message_hub_ptr
1490     );
1491
1492     this->has_improvement = true;
1493     this->__closeBuildMenu();
1494
1495     if (
```

```

1496         (this->tile_type == TileType :: LAKE) or
1497         (this->tile_type == TileType :: OCEAN)
1498     ) {
1499         this->decoration_cleared = true;
1500         this->assets_manager_ptr->getSound("splash")->play();
1501     }
1502
1503     this->__sendCreditsSpentMessage(build_cost);
1504     this->__sendTileStateMessage();
1505     this->__sendGameStateRequest();
1506
1507     return;
1508 } /* __buildWindTurbine() */

```

4.7.3.8 __clearDecoration()

```

void HexTile::__clearDecoration (
    void ) [private]

```

Helper method to clear tile decoration.

```

807 {
808     this->decoration_cleared = true;
809     this->draw_explosion = true;
810
811     switch (this->tile_type) {
812         case (TileType :: FOREST): {
813             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
814
815             break;
816         }
817
818         case (TileType :: MOUNTAINS): {
819             this->assets_manager_ptr->getSound("clear mountains tile")->play();
820
821             break;
822         }
823
824         case (TileType :: PLAINS): {
825             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
826
827             break;
828         }
829
830         default: {
831             // do nothing!
832
833             break;
834         }
835     }
836
837     return;
838 } /* __clearDecoration() */

```

4.7.3.9 __closeBuildMenu()

```

void HexTile::__closeBuildMenu (
    void ) [private]

```

Helper method to close the tile improvement build menu.

```

1299 {
1300     if (not this->build_menu_open) {
1301         return;
1302     }
1303
1304     this->build_menu_open = false;
1305     this->assets_manager_ptr->getSound("build menu close")->play();
1306
1307     return;
1308 } /* __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.

```
1718 {
1719     std::string coords_substring = "TILE COORDS: ";
1720     coords_substring += std::to_string(int(this->position_x - 400));
1721     coords_substring += ", ";
1722     coords_substring += std::to_string(int(this->position_y - 400));
1723     coords_substring += "\n";
1724
1725     return coords_substring;
1726 } /* __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.

```
1877 {
1878     std::string improvement_substring = "TILE IMPROVEMENT: ";
1879
1880     if (this->has_improvement) {
1881         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1882         improvement_substring += "\n";
1883     }
1884
1885     else {
1886         improvement_substring += "NONE\n";
1887     }
1888
1889     return improvement_substring;
1890 } /* __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.

```

1907 {
1908     //          32 char x 17 line console "-----\n";
1909     std::string options_substring          = "      **** TILE OPTIONS **** \n";
1910     options_substring                     += "                                     \n";
1911
1912     if (this->game_phase == "build settlement") {
1913         if (
1914             (this->tile_type != TileType :: OCEAN) and
1915             (this->tile_type != TileType :: LAKE)
1916         ) {
1917             options_substring += "[B]:  BUILD SETTLEMENT (";
1918             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
1919             options_substring += " K)\n";
1920         }
1921     }
1922
1923
1924     else if (this->game_phase == "system management") {
1925         if (this->has_improvement) {
1926             options_substring.clear();
1927             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
1928         }
1929
1930
1931         else if (not this->resource_assessed) {
1932             options_substring += "[A]:  ASSESS RESOURCE (";
1933             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
1934             options_substring += " K)\n";
1935         }
1936
1937
1938         else if (
1939             (not this->decoration_cleared) and
1940             (this->tile_type != TileType :: OCEAN) and
1941             (this->tile_type != TileType :: LAKE)
1942         ) {
1943             options_substring += "[C]:  CLEAR TILE (";
1944
1945             switch (this->tile_type) {
1946                 case (TileType :: FOREST): {
1947                     options_substring += std::to_string(CLEAR_FOREST_COST);
1948
1949                     break;
1950                 }
1951
1952
1953                 case (TileType :: MOUNTAINS): {
1954                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
1955
1956                     break;
1957                 }
1958
1959                 case (TileType :: PLAINS): {
1960                     options_substring += std::to_string(CLEAR_PLAINS_COST);
1961
1962                     break;
1963                 }
1964             }
1965
1966             default: {
1967                 //do nothing!
1968
1969                 break;
1970             }
1971         }
1972
1973         options_substring += " K)\n";
1974     }
1975
1976
1977     else if (
1978         (this->decoration_cleared) or
1979         (this->tile_type == TileType :: OCEAN) or
1980         (this->tile_type == TileType :: LAKE)
1981     ) {
1982         options_substring += "[B]:  OPEN BUILD MENU\n";
1983     }
1984
1985 }
1986
1987
1988     else if (this->game_phase == "victory") {
1989         options_substring          += "      **** VICTORY **** \n";
1990     }

```

```

1991
1992
1993     else {
1994         options_substring += "      **** LOSS ****      \n";
1995     }
1996
1997     return options_substring;
1998 } /* __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.

```

1807 {
1808     std::string resource_substring = "TILE RESOURCE:      ";
1809
1810     if (this->resource_assessed) {
1811         switch (this->tile_resource) {
1812             case (TileResource :: POOR): {
1813                 resource_substring += "POOR\n";
1814
1815                 break;
1816             }
1817
1818
1819             case (TileResource ::BELOW_AVERAGE): {
1820                 resource_substring += "BELOW AVERAGE\n";
1821
1822                 break;
1823             }
1824
1825
1826             case (TileResource :: AVERAGE): {
1827                 resource_substring += "AVERAGE\n";
1828
1829                 break;
1830             }
1831
1832
1833             case (TileResource :: ABOVE_AVERAGE): {
1834                 resource_substring += "ABOVE AVERAGE\n";
1835
1836                 break;
1837             }
1838
1839
1840             case (TileResource :: GOOD): {
1841                 resource_substring += "GOOD\n";
1842
1843                 break;
1844             }
1845
1846
1847             default: {
1848                 resource_substring += "???\n";
1849
1850                 break;
1851             }
1852         }
1853     }
1854
1855     else {
1856         resource_substring += "???\n";
1857     }
1858
1859     return resource_substring;
1860 } /* __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.

```
1743 {
1744     std::string type_substring = "TILE TYPE:      ";
1745
1746     switch (this->tile_type) {
1747         case (TileType :: FOREST): {
1748             type_substring += "FOREST\n";
1749
1750             break;
1751         }
1752
1753         case (TileType :: LAKE): {
1754             type_substring += "LAKE\n";
1755
1756             break;
1757         }
1758
1759         case (TileType :: MOUNTAINS): {
1760             type_substring += "MOUNTAINS\n";
1761
1762             break;
1763         }
1764
1765         case (TileType :: OCEAN): {
1766             type_substring += "OCEAN\n";
1767
1768             break;
1769         }
1770
1771         case (TileType :: PLAINS): {
1772             type_substring += "PLAINS\n";
1773
1774             break;
1775         }
1776
1777         default: {
1778             type_substring += "???\n";
1779
1780             break;
1781         }
1782     }
1783
1784     return type_substring;
1785 }
1786
1787 /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
890 {
891     if (not this->is_selected) {
892         return;
893     }
894
895     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
```



```
897         this->__setIsSelected(false);
898     }
899
900
901     if (this->build_menu_open) {
902         switch (this->tile_type) {
903             case (TileType :: FOREST): {
904                 switch (this->event_ptr->key.code) {
905                     case (sf::Keyboard::D): {
906                         this->__buildDieselGenerator();
907
908                         break;
909                     }
910
911                     case (sf::Keyboard::S): {
912                         this->__buildSolarPV();
913
914                         break;
915                     }
916
917                     case (sf::Keyboard::W): {
918                         this->__buildWindTurbine();
919
920                         break;
921                     }
922
923                     case (sf::Keyboard::E): {
924                         this->__buildEnergyStorage();
925
926                         break;
927                     }
928
929                     default: {
930                         // do nothing!
931
932                         break;
933                     }
934                 }
935             }
936
937             break;
938         }
939
940     }
941
942
943     case (TileType :: LAKE): {
944         switch (this->event_ptr->key.code) {
945             case (sf::Keyboard::S): {
946                 this->__buildSolarPV();
947
948                 break;
949             }
950
951             case (sf::Keyboard::W): {
952                 this->__buildWindTurbine();
953
954                 break;
955             }
956
957             default: {
958                 // do nothing!
959
960                 break;
961             }
962         }
963
964     }
965
966     break;
967 }
968
969
970     case (TileType :: MOUNTAINS): {
971         switch (this->event_ptr->key.code) {
972             case (sf::Keyboard::D): {
973                 this->__buildDieselGenerator();
974
975                 break;
976             }
977
978             case (sf::Keyboard::S): {
979                 this->__buildSolarPV();
980
981                 break;
982             }
983
984             break;
```

```

984         }
985
986
987         case (sf::Keyboard::W): {
988             this->__buildWindTurbine();
989
990             break;
991         }
992
993
994         case (sf::Keyboard::E): {
995             this->__buildEnergyStorage();
996
997             break;
998         }
999
1000
1001         default: {
1002             // do nothing!
1003
1004             break;
1005         }
1006     }
1007
1008     break;
1009 }
1010
1011
1012 case (TileType :: OCEAN): {
1013     switch (this->event_ptr->key.code) {
1014         case (sf::Keyboard::W): {
1015             this->__buildWindTurbine();
1016
1017             break;
1018         }
1019
1020
1021         case (sf::Keyboard::T): {
1022             this->__buildTidalTurbine();
1023
1024             break;
1025         }
1026
1027
1028         case (sf::Keyboard::A): {
1029             this->__buildWaveEnergyConverter();
1030
1031             break;
1032         }
1033
1034
1035         default: {
1036             // do nothing!
1037
1038             break;
1039         }
1040     }
1041
1042     break;
1043 }
1044
1045
1046 case (TileType :: PLAINS): {
1047     switch (this->event_ptr->key.code) {
1048         case (sf::Keyboard::D): {
1049             this->__buildDieselGenerator();
1050
1051             break;
1052         }
1053
1054
1055         case (sf::Keyboard::S): {
1056             this->__buildSolarPV();
1057
1058             break;
1059         }
1060
1061
1062         case (sf::Keyboard::W): {
1063             this->__buildWindTurbine();
1064
1065             break;
1066         }
1067
1068
1069         case (sf::Keyboard::E): {
1070             this->__buildEnergyStorage();

```

```

1071
1072             break;
1073         }
1074
1075         default: {
1076             // do nothing!
1077
1078             break;
1079         }
1080     }
1081 }
1082
1083 break;
1084 }
1085
1086 default: {
1087     //do nothing!
1088
1089     break;
1090 }
1091 }
1092 }
1093 }
1094
1095
1096 if (this->game_phase == "build settlement") {
1097     if (
1098         (this->tile_type != TileType :: OCEAN) and
1099         (this->tile_type != TileType :: LAKE)
1100     ) {
1101         if (this->event_ptr->key.code == sf::Keyboard::B) {
1102             this->__buildSettlement();
1103         }
1104     }
1105 }
1106
1107
1108 else if (this->game_phase == "system management") {
1109     if (this->has_improvement) {
1110         if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
1111     {
1112         if (this->event_ptr->key.code == sf::Keyboard::P) {
1113             this->__scrapImprovement();
1114         }
1115     }
1116     /*
1117     * All other inputs will be caught and handled by
1118     * this->tile_improvement_ptr->processEvent()
1119     */
1120 }
1121
1122
1123 else if (not this->resource_assessed) {
1124     if (this->event_ptr->key.code == sf::Keyboard::A) {
1125         if (this->credits < RESOURCE_ASSESSMENT_COST) {
1126             std::cout << "Cannot assess resource: insufficient credits (need "
1127                 << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
1128
1129             this->__sendInsufficientCreditsMessage();
1130         }
1131
1132         else {
1133             this->assess();
1134             this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1135             this->__sendTileStateMessage();
1136             this->__sendGameStateRequest();
1137         }
1138     }
1139 }
1140
1141
1142 else if (
1143     (not this->decoration_cleared) and
1144     (this->tile_type != TileType :: OCEAN) and
1145     (this->tile_type != TileType :: LAKE)
1146 ) {
1147     if (this->event_ptr->key.code == sf::Keyboard::C) {
1148         int clear_cost = 0;
1149
1150         switch (this->tile_type) {
1151             case (TileType :: FOREST): {
1152                 clear_cost = CLEAR_FOREST_COST;
1153
1154                 break;
1155             }
1156

```

```

1157
1158         case (TileType :: MOUNTAINS): {
1159             clear_cost = CLEAR_MOUNTAINS_COST;
1160
1161             break;
1162         }
1163
1164
1165         case (TileType :: PLAINS): {
1166             clear_cost = CLEAR_PLAINS_COST;
1167
1168             break;
1169         }
1170
1171
1172         default: {
1173             // do nothing!
1174
1175             break;
1176         }
1177     }
1178
1179     if (this->credits < clear_cost) {
1180         std::cout << "Cannot clear tile: insufficient credits (need "
1181             << clear_cost << " K)" << std::endl;
1182
1183         this->__sendInsufficientCreditsMessage();
1184     }
1185
1186     else {
1187         this->__clearDecoration();
1188         this->__sendCreditsSpentMessage(clear_cost);
1189         this->__sendTileStateMessage();
1190         this->__sendGameStateRequest();
1191     }
1192 }
1193
1194
1195
1196     else if (
1197         (this->decoration_cleared) or
1198         (this->tile_type == TileType :: OCEAN) or
1199         (this->tile_type == TileType :: LAKE)
1200     ) {
1201         if (this->event_ptr->key.code == sf::Keyboard::B) {
1202             this->__openBuildMenu();
1203         }
1204     }
1205 }
1206
1207     return;
1208 } /* __handleKeyPressEvents() */

```

4.7.3.16 __handleMouseButtonEvents()

```

void HexTile::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

1223 {
1224     switch (this->event_ptr->mouseButton.button) {
1225         case (sf::Mouse::Left): {
1226             if (this->__isClicked()) {
1227                 std::cout << "Tile (" << this->position_x << ", " <<
1228                     this->position_y << ") was selected" << std::endl;
1229
1230                 this->__setIsSelected(true);
1231
1232                 this->__sendTileSelectedMessage();
1233                 this->__sendTileStateMessage();
1234                 this->__sendGameStateRequest();
1235             }
1236
1237             else {
1238                 this->__setIsSelected(false);
1239             }
1240
1241             break;

```

```

1242     }
1243
1244
1245     case (sf::Mouse::Right): {
1246         this->__setIsSelected(false);
1247
1248         break;
1249     }
1250
1251
1252     default: {
1253         // do nothing!
1254
1255         break;
1256     }
1257 }
1258
1259 return;
1260 } /* __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.

```

858 {
859     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
860
861     double mouse_x = mouse_position.x;
862     double mouse_y = mouse_position.y;
863
864     double distance = sqrt(
865         pow(this->position_x - mouse_x, 2) +
866         pow(this->position_y - mouse_y, 2)
867     );
868
869     if (distance < this->minor_radius) {
870         return true;
871     }
872     else {
873         return false;
874     }
875 } /* __isClicked() */

```

4.7.3.18 __openBuildMenu()

```

void HexTile::__openBuildMenu (
    void ) [private]

```

Helper method to open the tile improvement build menu.

```

1275 {
1276     if (this->build_menu_open) {
1277         return;
1278     }
1279
1280     this->build_menu_open = true;
1281     this->assets_manager_ptr->getSound("build menu open")->play();
1282
1283     return;
1284 } /* __openBuildMenu() */

```

4.7.3.19 __scrapImprovement()

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

Helper method to scrap the tile improvement ([Settlement](#) cannot be scrapped).

```
1656 {
1657     this->draw_explosion = true;
1658     this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1659
1660     delete this->tile_improvement_ptr;
1661     this->tile_improvement_ptr = NULL;
1662
1663     this->has_improvement = false;
1664
1665     if (
1666         (this->tile_type == TileType :: LAKE) or
1667         (this->tile_type == TileType :: OCEAN)
1668     ) {
1669         this->decoration_cleared = false;
1670     }
1671
1672     this->__sendCreditsSpentMessage(SCRAP_COST);
1673     this->__sendTileStateMessage();
1674     this->__sendGameStateRequest();
1675
1676     return;
1677 } /* __scrapImprovement() */
```

4.7.3.20 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2055 {
2056     Message assess_neighbours_message;
2057
2058     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2059     assess_neighbours_message.subject = "assess neighbours";
2060
2061     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2062
2063     std::cout << "Assess neighbours message sent by " << this << std::endl;
2064
2065     return;
2066 } /* __sendAssessNeighboursMessage() */
```

4.7.3.21 __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.
----------------------	--

```
2138 {
2139     Message credits_spent_message;
```

```

2140
2141     credits_spent_message.channel = GAME_CHANNEL;
2142     credits_spent_message.subject = "credits spent";
2143
2144     credits_spent_message.int_payload["credits spent"] = credits_spent;
2145
2146     this->message_hub_ptr->sendMessage(credits_spent_message);
2147
2148     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2149         << std::endl;
2150     return;
2151 } /* __sendCreditsSpentMessage() */

```

4.7.3.22 __sendGameStateRequest()

```

void HexTile::__sendGameStateRequest (
    void ) [private]

```

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

```

2081 {
2082     Message game_state_request;
2083
2084     game_state_request.channel = GAME_CHANNEL;
2085     game_state_request.subject = "state request";
2086
2087     this->message_hub_ptr->sendMessage(game_state_request);
2088
2089     std::cout << "Game state request message sent by " << this << std::endl;
2090     return;
2091 } /* __sendGameStateRequest() */

```

4.7.3.23 __sendInsufficientCreditsMessage()

```

void HexTile::__sendInsufficientCreditsMessage (
    void ) [private]

```

Helper method to format and send an insufficient credits message.

```

2166 {
2167     Message insufficient_credits_message;
2168
2169     insufficient_credits_message.channel = GAME_CHANNEL;
2170     insufficient_credits_message.subject = "insufficient credits";
2171
2172     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2173
2174     std::cout << "Insufficient credits message sent by " << this << std::endl;
2175     return;
2176
2177 } /* __sendInsufficientCreditsMessage() */

```

4.7.3.24 __sendTileSelectedMessage()

```

void HexTile::__sendTileSelectedMessage (
    void ) [private]

```

Helper method to format and send message on tile selection.

```

1692 {
1693     Message tile_selected_message;
1694
1695     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1696     tile_selected_message.subject = "tile selected";
1697
1698     this->message_hub_ptr->sendMessage(tile_selected_message);
1699
1700     return;
1701 } /* __sendTileSelectedMessage() */

```

4.7.3.25 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2013 {
2014     Message tile_state_message;
2015
2016     tile_state_message.channel = TILE_STATE_CHANNEL;
2017     tile_state_message.subject = "tile state";
2018
2019
2020     //          32 char x 17 line console "-----\n";
2021     std::string console_string = "      *** TILE INFO ***\n";
2022     console_string += "\n";
2023
2024     console_string += this->__getTileCoordsSubstring();
2025     console_string += "\n";
2026
2027     console_string += this->__getTileTypeSubstring();
2028     console_string += this->__getTileResourceSubstring();
2029     console_string += this->__getTileImprovementSubstring();
2030     console_string += "\n";
2031
2032     console_string += this->__getTileOptionsSubstring();
2033
2034     tile_state_message.string_payload["console string"] = console_string;
2035
2036     this->message_hub_ptr->sendMessage(tile_state_message);
2037
2038     std::cout << "Tile state message sent by " << this << std::endl;
2039     return;
2040 } /* __sendTileStateMessage() */
```

4.7.3.26 __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.
-------------------	-------------------------

```
2108 {
2109     Message update_game_phase_message;
2110
2111     update_game_phase_message.channel = GAME_CHANNEL;
2112     update_game_phase_message.subject = "update game phase";
2113
2114     update_game_phase_message.string_payload["game phase"] = game_phase;
2115
2116     this->message_hub_ptr->sendMessage(update_game_phase_message);
2117
2118     std::cout << "Update game phase message sent by " << this << std::endl;
2119
2120     return;
2121 } /* __sendUpdateGamePhaseMessage() */
```

4.7.3.27 __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.
--------------------	--

```

763 {
764     this->is_selected = is_selected;
765
766     if (this->tile_improvement_ptr != NULL) {
767         this->tile_improvement_ptr->setIsSelected(is_selected);
768
769         if (is_selected) {
770             switch (this->tile_improvement_ptr->tile_improvement_type) {
771                 case (TileImprovementType :: SETTLEMENT): {
772                     this->assets_manager_ptr->getSound("people and children")->play();
773
774                     break;
775                 }
776
777                 default: {
778                     // do nothing!
779
780                     break;
781                 }
782             }
783         }
784     }
785 }
786
787 if ((not is_selected) and this->build_menu_open) {
788     this->__closeBuildMenu();
789 }
790
791 return;
792 } /* __setIsSelected() */

```

4.7.3.28 __setResourceText()

```

void HexTile::__setResourceText (
    void ) [private]

```

Helper method to set up resource text.

```

193 {
194     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
195
196     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
197
198     if (this->resource_assessed) {
199         switch (this->tile_resource) {
200             case (TileResource :: POOR): {
201                 this->resource_text.setString("-2");
202                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
203
204                 break;
205             }
206
207             case (TileResource :: BELOW_AVERAGE): {
208                 this->resource_text.setString("-1");
209                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
210
211                 break;
212             }
213
214             case (TileResource :: AVERAGE): {
215                 this->resource_text.setString("+0");
216
217                 break;
218             }
219
220             case (TileResource :: ABOVE_AVERAGE): {
221                 this->resource_text.setString("+1");
222                 this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
223
224                 break;

```

```

225         }
226
227         case (TileResource :: GOOD): {
228             this->resource_text.setString("+2");
229             this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
230
231             break;
232         }
233
234         default: {
235             this->resource_text.setString("");
236
237             break;
238         }
239     }
240 }
241
242 else {
243     this->resource_text.setString("");
244 }
245
246 this->resource_text.setCharacterSize(20);
247
248 this->resource_text.setOrigin(
249     this->resource_text.getLocalBounds().width / 2,
250     this->resource_text.getLocalBounds().height / 2
251 );
252
253 this->resource_text.setPosition(
254     this->position_x,
255     this->position_y - 4
256 );
257
258 this->resource_text.setOutlineThickness(1);
259 this->resource_text.setOutlineColor(sf::Color(0, 0, 0, 255));
260
261 return;
262 } /* __setResourceText() */

```

4.7.3.29 __setUpBuildMenu()

```

void HexTile::__setUpBuildMenu (
    void ) [private]

```

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

```

666 {
667     this->build_menu_options_vec.clear();
668     this->build_menu_options_text_vec.clear();
669
670     // 1. set up and place build menu backing and text
671     this->build_menu_backing.setSize(sf::Vector2f(600, 256));
672     this->build_menu_backing.setOrigin(300, 128);
673     this->build_menu_backing.setPosition(400, 400);
674     this->build_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
675     this->build_menu_backing.setOutlineColor(MENU_FRAME_GREY);
676     this->build_menu_backing.setOutlineThickness(4);
677
678     this->build_menu_backing_text.setString("**** BUILD MENU ****");
679     this->build_menu_backing_text.setFont(
680         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
681     );
682     this->build_menu_backing_text.setCharacterSize(16);
683     this->build_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
684     this->build_menu_backing_text.setOrigin(
685         this->build_menu_backing_text.getLocalBounds().width / 2, 0
686     );
687     this->build_menu_backing_text.setPosition(400, 400 - 128 + 4);
688
689     // 2. set up and place build menu option sprites and text
690     switch (this->tile_type) {
691         case (TileType :: FOREST): {
692             this->__setUpDieselGeneratorBuildOption();
693             this->__setUpSolarPVBuildOption();
694             this->__setUpWindTurbineBuildOption();
695             this->__setUpEnergyStorageSystemBuildOption();
696
697             break;
698         }

```

```

699
700
701     case (TileType :: LAKE): {
702         this->__setUpSolarPVBuildOption(true);
703         this->__setUpWindTurbineBuildOption(true);
704
705         break;
706     }
707
708
709     case (TileType :: MOUNTAINS): {
710         this->__setUpDieselGeneratorBuildOption();
711         this->__setUpSolarPVBuildOption();
712         this->__setUpWindTurbineBuildOption();
713         this->__setUpEnergyStorageSystemBuildOption();
714
715         break;
716     }
717
718
719     case (TileType :: OCEAN): {
720         this->__setUpWindTurbineBuildOption(false, true);
721         this->__setUpTidalTurbineBuildOption();
722         this->__setUpWaveEnergyConverterBuildOption();
723
724         break;
725     }
726
727
728     case (TileType :: PLAINS): {
729         this->__setUpDieselGeneratorBuildOption();
730         this->__setUpSolarPVBuildOption();
731         this->__setUpWindTurbineBuildOption();
732         this->__setUpEnergyStorageSystemBuildOption();
733
734         break;
735     }
736
737
738     default: {
739         // do nothing!
740
741         break;
742     }
743 }
744
745 return;
746 } /* __setUpBuildMenu() */

```

4.7.3.30 __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.

```

357 {
358     size_t n_options = this->build_menu_options_vec.size();
359
360     // 1. set up option sprite(s)
361     this->build_menu_options_vec.push_back({});
362
363     if (not texture_key.empty()) {
364         sf::Sprite texture_sheet(
365             *(this->assets_manager_ptr->getTexture(texture_key))
366         );
367

```

```

368         int sheet_height = texture_sheet.getLocalBounds().height;
369         int n_subrects = sheet_height / 64;
370
371         for (int i = 0; i < n_subrects; i++) {
372             this->build_menu_options_vec.back().push_back(
373                 sf::Sprite(
374                     *(this->assets_manager_ptr->getTexture(texture_key)),
375                     sf::IntRect(0, i * 64, 64, 64)
376                 )
377             );
378
379             this->build_menu_options_vec.back().back().setOrigin(
380                 this->build_menu_options_vec.back().back().getLocalBounds().width / 2,
381                 this->build_menu_options_vec.back().back().getLocalBounds().height
382             );
383
384             this->build_menu_options_vec.back().back().setPosition(
385                 400 - 300 + 75 + n_options * 150,
386                 400 - 32
387             );
388         }
389     }
390
391     else {
392         this->build_menu_options_vec.back().push_back(sf::Sprite());
393     }
394
395
396     // 2. set up option text
397     this->build_menu_options_text_vec.push_back(
398         sf::Text(
399             option_string,
400             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
401             16
402         )
403     );
404
405     this->build_menu_options_text_vec.back().setOrigin(
406         this->build_menu_options_text_vec.back().getLocalBounds().width / 2,
407         0
408     );
409
410     this->build_menu_options_text_vec.back().setPosition(
411         400 - 300 + 75 + n_options * 150,
412         400 - 16 - 4
413     );
414
415     this->build_menu_options_text_vec.back().setFillColor(MONOCROME_TEXT_GREEN);
416
417     return;
418 } /* __setUpBuildOption() */

```

4.7.3.31 __setUpDieselGeneratorBuildOption()

```

void HexTile::__setUpDieselGeneratorBuildOption (
    void ) [private]

```

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

```

433 {
434     // 1. set up option sprite(s)
435     std::string texture_key = "diesel generator";
436
437     // 2. set up option string (up to 16 chars wide)
438     // -----\n"
439     std::string diesel_generator_string = "DIESEL GENERATOR\n";
440     diesel_generator_string += "          \n";
441     diesel_generator_string += "CAPACITY: 100 kW\n";
442     diesel_generator_string += "COST:      ";
443     diesel_generator_string += std::to_string(DIESEL_GENERATOR_BUILD_COST);
444     diesel_generator_string += " K\n\n";
445     diesel_generator_string += "BUILD:    [D]   \n";
446
447     // 3. call general method
448     this->__setUpBuildOption(texture_key, diesel_generator_string);
449
450     return;
451 } /* __setUpDieselGeneratorBuildOption() */

```

4.7.3.32 __setUpEnergyStorageSystemBuildOption()

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

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

```
633 {
634     // 1. set up option sprite(s)
635     std::string texture_key = "energy storage system";
636
637     // 2. set up option string (up to 16 chars wide)
638     // -----\n"
639     std::string energy_storage_system_string = " ENERGY STORAGE \n";
640     energy_storage_system_string += " \n";
641     energy_storage_system_string += "CAPCTY: 500 kWh\n";
642     energy_storage_system_string += "COST: ";
643     energy_storage_system_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
644     energy_storage_system_string += " K\n\n";
645     energy_storage_system_string += "BUILD: [E] \n";
646
647     // 3. call general method
648     this->__setUpBuildOption(texture_key, energy_storage_system_string);
649
650     return;
651 } /* __setUpEnergyStorageSystemBuildOption() */
```

4.7.3.33 __setUpMagnifyingGlassSprite()

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

Helper method to set up and position magnifying glass sprite.

```
277 {
278     this->magnifying_glass_sprite.setTexture(
279         *(this->assets_manager_ptr->getTexture("magnifying_glass_64x64_1"))
280     );
281
282     this->magnifying_glass_sprite.setOrigin(
283         this->magnifying_glass_sprite.getLocalBounds().width / 2,
284         this->magnifying_glass_sprite.getLocalBounds().height / 2
285     );
286
287     this->magnifying_glass_sprite.setPosition(
288         this->position_x,
289         this->position_y
290     );
291
292     return;
293 } /* __setUpMagnifyingGlassSprite() */
```

4.7.3.34 __setUpNodeSprite()

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

Helper method to set up node sprite.

```
68 {
69     this->node_sprite.setRadius(4);
70
71     this->node_sprite.setOrigin(
72         this->node_sprite.getLocalBounds().width / 2,
73         this->node_sprite.getLocalBounds().height / 2
74     );
75
76     this->node_sprite.setPosition(this->position_x, this->position_y);
77
78     this->node_sprite.setFillColor(sf::Color(255, 0, 0, 255));
79
80     return;
81 } /* __setUpNodeSprite() */
```

4.7.3.35 __setUpResourceChipSprite()

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

Helper method to set up resource chip sprite.

```
166 {
167     this->resource_chip_sprite.setRadius(2 * this->minor_radius / 3);
168
169     this->resource_chip_sprite.setOrigin(
170         this->resource_chip_sprite.getLocalBounds().width / 2,
171         this->resource_chip_sprite.getLocalBounds().height / 2
172     );
173
174     this->resource_chip_sprite.setPosition(this->position_x, this->position_y);
175
176     this->resource_chip_sprite.setFillColor(RESOURCE_CHIP_GREY);
177
178     return;
179 } /* __setUpResourceChip() */
```

4.7.3.36 __setUpSelectOutlineSprite()

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

Helper method to set up select outline sprite.

```
130 {
131     int n_points = 6;
132
133     this->select_outline_sprite.setPointCount(n_points);
134
135     for (int i = 0; i < n_points; i++) {
136         this->select_outline_sprite.setPoint(
137             i,
138             sf::Vector2f(
139                 this->position_x + this->major_radius * cos((30 + 60 * i) * (M_PI / 180)),
140                 this->position_y + this->major_radius * sin((30 + 60 * i) * (M_PI / 180))
141             )
142         );
143     }
144
145     this->select_outline_sprite.setOutlineThickness(4);
146     this->select_outline_sprite.setOutlineColor(MONOCHROME_TEXT_RED);
147
148     this->select_outline_sprite.setFillColor(sf::Color(0, 0, 0, 0));
149
150     return;
151 } /* __setUpSelectOutline() */
```

4.7.3.37 __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.
----------------	---------------------------

```

521 {
522     // 1. set up option sprite(s)
523     std::string texture_key = "solar PV array";
524
525     // 2. set up option string (up to 16 chars wide)
526     int build_cost = SOLAR_PV_BUILD_COST;
527     if (is_lake) {
528         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
529     }
530
531     //
532     std::string solar_PV_string = "-----\n"
533     solar_PV_string += " SOLAR PV ARRAY \n";
534     solar_PV_string += "CAPACITY: 100 kW\n";
535     solar_PV_string += "COST: ";
536     solar_PV_string += std::to_string(build_cost);
537     solar_PV_string += " K";
538
539     if (is_lake) {
540         solar_PV_string += "\n** LAKE BUILD **\n\n";
541     }
542     else {
543         solar_PV_string += "\n\n\n";
544     }
545
546     solar_PV_string += "BUILD: [S] \n";
547
548     // 3. call general method
549     this->__setUpBuildOption(texture_key, solar_PV_string);
550
551     return;
552 } /* __setUpSolarPVBuildOption() */

```

4.7.3.38 __setUpTidalTurbineBuildOption()

```

void HexTile::__setUpTidalTurbineBuildOption (
    void ) [private]

```

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

```

567 {
568     // 1. set up option sprite(s)
569     std::string texture_key = "tidal turbine";
570
571     // 2. set up option string (up to 16 chars wide)
572     //
573     std::string tidal_turbine_string = "-----\n"
574     tidal_turbine_string += " TIDAL TURBINE \n";
575     tidal_turbine_string += "CAPACITY: 100 kW\n";
576     tidal_turbine_string += "COST: ";
577     tidal_turbine_string += std::to_string(TIDAL_TURBINE_BUILD_COST);
578     tidal_turbine_string += " K\n\n";
579     tidal_turbine_string += "BUILD: [T] \n";
580
581     // 3. call general method
582     this->__setUpBuildOption(texture_key, tidal_turbine_string);
583
584     return;
585 } /* __setUpTidalTurbineBuildOption() */

```

4.7.3.39 __setUpTileExplosionReel()

```

void HexTile::__setUpTileExplosionReel (
    void ) [private]

```

Helper method to set up tile explosion sprite reel.

```

308 {
309     for (int i = 0; i < 4; i++) {
310         for (int j = 0; j < 4; j++) {
311             this->explosion_sprite_reel.push_back(

```

```

312         sf::Sprite(
313             *(this->assets_manager_ptr->getTexture("tile clear explosion")),
314             sf::IntRect(j * 64, i * 64, 64, 64)
315         )
316     );
317
318     this->explosion_sprite_reel.back().setOrigin(
319         this->explosion_sprite_reel.back().getLocalBounds().width / 2,
320         this->explosion_sprite_reel.back().getLocalBounds().height / 2
321     );
322
323     this->explosion_sprite_reel.back().setPosition(
324         this->position_x,
325         this->position_y
326     );
327 }
328 }
329
330 return;
331 } /* __setUpTileExplosionReel() */

```

4.7.3.40 __setUpTileSprite()

```

void HexTile::__setUpTileSprite (
    void ) [private]

```

Helper method to set up tile sprite.

```

96 {
97     int n_points = 6;
98
99     this->tile_sprite.setPointCount(n_points);
100
101     for (int i = 0; i < n_points; i++) {
102         this->tile_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->tile_sprite.setOutlineThickness(1);
112     this->tile_sprite.setOutlineColor(sf::Color(175, 175, 175, 255));
113
114     return;
115 } /* __setUpTileSprite() */

```

4.7.3.41 __setUpWaveEnergyConverterBuildOption()

```

void HexTile::__setUpWaveEnergyConverterBuildOption (
    void ) [private]

```

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

```

600 {
601     // 1. set up option sprite(s)
602     std::string texture_key = "wave energy converter";
603
604     // 2. set up option string (up to 16 chars wide)
605     // -----\n"
606     std::string wave_energy_converter_string = "WAVE ENERGY CVTR\n";
607     wave_energy_converter_string += " \n";
608     wave_energy_converter_string += "CAPACITY: 100 kW\n";
609     wave_energy_converter_string += "COST: ";
610     wave_energy_converter_string += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
611     wave_energy_converter_string += " K\n\n";
612     wave_energy_converter_string += "BUILD: [A] \n";
613
614     // 3. call general method
615     this->__setUpBuildOption(texture_key, wave_energy_converter_string);
616
617     return;
618 } /* __setUpWaveEnergyConverterBuildOption() */

```


4.7.3.42 __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.

```
470 {
471     // 1. set up option sprite(s)
472     std::string texture_key = "wind turbine";
473
474     // 2. set up option string (up to 16 chars wide)
475     int build_cost = WIND_TURBINE_BUILD_COST;
476     if (is_lake or is_ocean) {
477         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
478     }
479
480     // ----- \n"
481     std::string wind_turbine_string = " WIND TURBINE \n";
482     wind_turbine_string += " \n";
483     wind_turbine_string += "CAPACITY: 100 kW\n";
484     wind_turbine_string += "COST: ";
485     wind_turbine_string += std::to_string(build_cost);
486     wind_turbine_string += " K";
487
488     if (is_lake) {
489         wind_turbine_string += "\n* LAKE BUILD *\n\n";
490     }
491     else if (is_ocean) {
492         wind_turbine_string += "\n* OCEAN BUILD * \n\n";
493     }
494     else {
495         wind_turbine_string += "\n\n\n";
496     }
497
498     wind_turbine_string += "BUILD: [W] \n";
499
500     // 3. call general method
501     this->__setUpBuildOption(texture_key, wind_turbine_string);
502
503     return;
504 } /* __setUpWindTurbineBuildOption() */
```

4.7.3.43 assess()

```
void HexTile::assess (
    void )
```

Method to assess the tile's resource.

```
2598 {
2599     this->resource_assessed = true;
2600     this->resource_assessment = true;
2601
2602     this->assets_manager_ptr->getSound("resource assessment")->play();
2603
2604     this->__setResourceText();
2605     this->__sendTileStateMessage();
2606
2607     return;
2608 } /* assess() */
```

4.7.3.44 decorateTile()

```
void HexTile::decorateTile (
    void )
```

Method to decorate tile.

```
2476 {
2477     switch (this->tile_type) {
2478     case (TileType :: FOREST): {
2479         this->tile_decoration_sprite.setTexture(
2480             *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2481         );
2482
2483         break;
2484     }
2485
2486     case (TileType :: LAKE): {
2487         this->tile_decoration_sprite.setTexture(
2488             *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2489         );
2490
2491         break;
2492     }
2493
2494     case (TileType :: MOUNTAINS): {
2495         this->tile_decoration_sprite.setTexture(
2496             *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2497         );
2498
2499         break;
2500     }
2501
2502     case (TileType :: OCEAN): {
2503         this->tile_decoration_sprite.setTexture(
2504             *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2505         );
2506
2507         break;
2508     }
2509
2510     case (TileType :: PLAINS): {
2511         this->tile_decoration_sprite.setTexture(
2512             *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2513         );
2514
2515         break;
2516     }
2517
2518     default: {
2519         // do nothing!
2520
2521         break;
2522     }
2523 }
2524
2525
2526 if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
2527     this->tile_decoration_sprite.setOrigin(
2528         this->tile_decoration_sprite.getLocalBounds().width / 2,
2529         this->tile_decoration_sprite.getLocalBounds().height / 2
2530     );
2531
2532     this->tile_decoration_sprite.setPosition(
2533         this->position_x,
2534         this->position_y
2535     );
2536
2537     if ((double)rand() / RAND_MAX > 0.5) {
2538         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2539     }
2540 }
2541
2542 else {
2543     this->tile_decoration_sprite.setOrigin(
2544         this->tile_decoration_sprite.getLocalBounds().width / 2,
2545         this->tile_decoration_sprite.getLocalBounds().height
2546     );
2547
2548     this->tile_decoration_sprite.setPosition(
2549         this->position_x,
2550         this->position_y + 12
2551     );
2552
2553     if ((double)rand() / RAND_MAX > 0.5) {
```

```

2554         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2555     }
2556 }
2557
2558     return;
2559 } /* decorateTile(void) */

```

4.7.3.45 draw()

```

void HexTile::draw (
    void )

```

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

```

2703 {
2704     // 1. draw hex
2705     this->render_window_ptr->draw(this->tile_sprite);
2706
2707     // 2. draw node
2708     if (this->show_node) {
2709         this->render_window_ptr->draw(this->node_sprite);
2710     }
2711
2712     // 3. draw tile decoration
2713     if (not this->decoration_cleared) {
2714         this->render_window_ptr->draw(this->tile_decoration_sprite);
2715     }
2716
2717     // 4. draw tile improvement
2718     if (this->has_improvement) {
2719         if (not this->tile_improvement_ptr->just_built) {
2720             this->tile_improvement_ptr->draw();
2721         }
2722     }
2723
2724     // 5. draw resource
2725     if (this->show_resource) {
2726         this->render_window_ptr->draw(this->resource_chip_sprite);
2727         this->render_window_ptr->draw(this->resource_text);
2728     }
2729
2730     // 6. draw selection outline
2731     if (this->is_selected) {
2732         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2733
2734         outline_colour.a =
2735             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2736
2737         this->select_outline_sprite.setOutlineColor(outline_colour);
2738
2739         this->render_window_ptr->draw(this->select_outline_sprite);
2740     }
2741
2742     // 7. draw resource assessment notification
2743     if (this->resource_assessment) {
2744         int alpha = this->magnifying_glass_sprite.getColor().a;
2745
2746         alpha -= 0.05 * FRAMES_PER_SECOND;
2747         if (alpha < 0) {
2748             alpha = 0;
2749             this->resource_assessment = false;
2750         }
2751
2752         this->magnifying_glass_sprite.setColor(
2753             sf::Color(255, 255, 255, alpha)
2754         );
2755
2756         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2757     }
2758
2759     // 8. draw explosion, then settlement placement
2760     if (this->draw_explosion) {
2761         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2762
2763         if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2764             this->explosion_frame++;
2765         }
2766
2767         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {

```

```

2768         this->draw_explosion = false;
2769         this->explosion_frame = 0;
2770     }
2771 }
2772
2773 else if (this->has_improvement) {
2774     if (this->tile_improvement_ptr->just_built) {
2775         this->tile_improvement_ptr->draw();
2776     }
2777 }
2778
2779 // 9. build menu
2780 if (this->build_menu_open) {
2781     this->render_window_ptr->draw(this->build_menu_backing);
2782     this->render_window_ptr->draw(this->build_menu_backing_text);
2783
2784     for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2785         for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2786             this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2787         }
2788         this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2789     }
2790 }
2791
2792 this->frame++;
2793 return;
2794 } /* draw() */

```

4.7.3.46 processEvent()

```

void HexTile::processEvent (
    void )

```

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

```

2623 {
2624     // 1. process TileImprovement events
2625     if (this->tile_improvement_ptr != NULL) {
2626         this->tile_improvement_ptr->processEvent();
2627     }
2628
2629     // 2. process HexTile events
2630     if (this->event_ptr->type == sf::Event::KeyPressed) {
2631         this->__handleKeyPressEvents();
2632     }
2633
2634     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2635         this->__handleMouseButtonEvents();
2636     }
2637
2638     return;
2639 } /* processEvent() */

```

4.7.3.47 processMessage()

```

void HexTile::processMessage (
    void )

```

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

```

2654 {
2655     // 1. process TileImprovement messages
2656     if (this->tile_improvement_ptr != NULL) {
2657         this->tile_improvement_ptr->processMessage();
2658     }
2659
2660     // 2. process HexTile messages
2661     if (this->is_selected) {
2662         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2663             Message game_state_message = this->message_hub_ptr->receiveMessage(
2664                 GAME_STATE_CHANNEL

```

```

2665         );
2666
2667         if (game_state_message.subject == "game state") {
2668             this->credits = game_state_message.int_payload["credits"];
2669             this->game_phase = game_state_message.string_payload["game phase"];
2670
2671             if (this->tile_improvement_ptr != NULL) {
2672                 this->tile_improvement_ptr->credits = this->credits;
2673                 this->tile_improvement_ptr->game_phase = this->game_phase;
2674             }
2675
2676             std::cout << "Game state message received by " << this << std::endl;
2677             this->__sendTileStateMessage();
2678             this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
2679         }
2680     }
2681
2682     std::cout << "Current credits (HexTile): " << this->credits << " K" <<
2683         std::endl;
2684 }
2685
2686 return;
2687 } /* processMessage() */

```

4.7.3.48 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].
--------------------	--

```

2425 {
2426     // 1. check input
2427     if (input_value < 0 or input_value > 1) {
2428         std::string error_str = "ERROR HexTile::setTileResource() given input value is ";
2429         error_str += "not in the closed interval [0, 1]";
2430
2431         #ifdef _WIN32
2432             std::cout << error_str << std::endl;
2433         #endif /* _WIN32 */
2434
2435         throw std::runtime_error(error_str);
2436     }
2437
2438     // 2. convert input value to tile resource
2439     TileResource tile_resource;
2440
2441     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2442         tile_resource = TileResource :: POOR;
2443     }
2444     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2445         tile_resource = TileResource :: BELOW_AVERAGE;
2446     }
2447     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2448         tile_resource = TileResource :: AVERAGE;
2449     }
2450     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2451         tile_resource = TileResource :: ABOVE_AVERAGE;
2452     }
2453     else {
2454         tile_resource = TileResource :: GOOD;
2455     }
2456
2457     // 3. call alternate method
2458     this->setTileResource(tile_resource);
2459
2460     return;
2461 } /* setTileResource(double) */

```

4.7.3.49 setTitleResource() [2/2]

```
void HexTile::setTitleResource (
    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.
----------------------	---

```
2403 {
2404     this->tile_resource = tile_resource;
2405     this->__setResourceText();
2406
2407     return;
2408 } /* setTitleResource(TileResource) */
```

4.7.3.50 setTitleType() [1/2]

```
void HexTile::setTitleType (
    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].
--------------------	--

```
2353 {
2354     // 1. check input
2355     if (input_value < 0 or input_value > 1) {
2356         std::string error_str = "ERROR HexTile::setTitleType() given input value is ";
2357         error_str += "not in the closed interval [0, 1]";
2358
2359         #ifdef _WIN32
2360             std::cout << error_str << std::endl;
2361         #endif /* _WIN32 */
2362
2363         throw std::runtime_error(error_str);
2364     }
2365
2366     // 2. convert input value to tile type
2367     TileType tile_type;
2368
2369     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2370         tile_type = TileType :: LAKE;
2371     }
2372     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2373         tile_type = TileType :: PLAINS;
2374     }
2375     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2376         tile_type = TileType :: FOREST;
2377     }
2378     else {
2379         tile_type = TileType :: MOUNTAINS;
2380     }
2381
2382     // 3. call alternate method
2383     this->setTileType(tile_type);
2384
2385     return;
2386 } /* setTitleType(double) */
```

4.7.3.51 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.
------------------	---

```
2292 {
2293     this->tile_type = tile_type;
2294
2295     switch (this->tile_type) {
2296     case (TileType :: FOREST): {
2297         this->tile_sprite.setFillColor(FOREST_GREEN);
2298
2299         break;
2300     }
2301
2302     case (TileType :: LAKE): {
2303         this->tile_sprite.setFillColor(LAKE_BLUE);
2304
2305         break;
2306     }
2307
2308     case (TileType :: MOUNTAINS): {
2309         this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2310
2311         break;
2312     }
2313
2314     case (TileType :: OCEAN): {
2315         this->tile_sprite.setFillColor(OCEAN_BLUE);
2316
2317         break;
2318     }
2319
2320     case (TileType :: PLAINS): {
2321         this->tile_sprite.setFillColor(PLAINS_YELLOW);
2322
2323         break;
2324     }
2325
2326     default: {
2327         // do nothing!
2328
2329         break;
2330     }
2331 }
2332
2333 this->__setUpBuildMenu();
2334
2335 return;
2336 } /* setTileType(TileType) */
```

4.7.3.52 toggleResourceOverlay()

```
void HexTile::toggleResourceOverlay (
    void )
```

Method to toggle the tile resource overlay.

```
2574 {
2575     if (this->show_resource) {
2576         this->show_resource = false;
2577     }
2578     else {
2579         this->show_resource = true;
2580     }
2581
2582     return;
2583 } /* 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

```
unsigned long long 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.

```
78 {
79     //...
80
81     std::cout << "MessageHub constructed at " << this << std::endl;
82
83     return;
84 } /* MessageHub() */
```


4.9.2.2 ~MessageHub()

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

Destructor for the [MessageHub](#) class.

```
425 {
426     this->clear();
427
428     std::cout << "MessageHub at " << this << " destroyed" << std::endl;
429
430     return;
431 } /* ~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.
----------------	--

```
129 {
130     // 1. check if channel is in map (if so, throw error)
131     if (this->message_map.count(channel) > 0) {
132         std::string error_str = "ERROR MessageHub::addChannel() channel ";
133         error_str += channel;
134         error_str += " is already in message map";
135
136         #ifdef _WIN32
137             std::cout << error_str << std::endl;
138         #endif /* _WIN32 */
139
140         throw std::runtime_error(error_str);
141     }
142
143     // 2. add channel to map
144     this->message_map[channel] = {};
145
146     std::cout << "Channel " << channel << " added to message hub" << std::endl;
147
148     return;
149 } /* addChannel() */
```

4.9.3.2 clear()

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

Method to clear the [MessageHub](#).

```
405 {
406
407     this->clearMessages();
```

```

408     this->message_map.clear();
409
410     return;
411 } /* clear() */

```

4.9.3.3 clearMessages()

```

void MessageHub::clearMessages (
    void )

```

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

```

379 {
380     std::map<std::string, std::list<Message>::iterator map_iter;
381     for (
382         map_iter = this->message_map.begin();
383         map_iter != this->message_map.end();
384         map_iter++
385     ) {
386         map_iter->second.clear();
387     }
388
389     return;
390 } /* clearMessages() */

```

4.9.3.4 hasTraffic()

```

bool MessageHub::hasTraffic (
    void )

```

Method to determine if there remains any message traffic.

```

99 {
100     std::map<std::string, std::list<Message>::iterator map_iter;
101     for (
102         map_iter = this->message_map.begin();
103         map_iter != this->message_map.end();
104         map_iter++
105     ) {
106         if (not map_iter->second.empty()) {
107             return true;
108         }
109     }
110
111     return false;
112 } /* 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.

```

244 {
245     // 1. check if channel is in map (if not, throw error)
246     if (this->message_map.count(channel) <= 0) {
247         std::string error_str = "ERROR MessageHub::isEmpty() channel ";
248         error_str += channel;
249         error_str += " is not in message map";
250
251         #ifdef _WIN32
252             std::cout << error_str << std::endl;
253         #endif /* _WIN32 */
254
255         throw std::runtime_error(error_str);
256     }
257
258     if (this->message_map[channel].empty()) {
259         return true;
260     }
261     else {
262         return false;
263     }
264 } /* 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.
----------------	---

```

333 {
334     // 1. check if channel is in map (if not, throw error)
335     if (this->message_map.count(channel) <= 0) {
336         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
337         error_str += channel;
338         error_str += " is not in message map";
339
340         #ifdef _WIN32
341             std::cout << error_str << std::endl;
342         #endif /* _WIN32 */
343
344         throw std::runtime_error(error_str);
345     }
346
347     // 2. check if channel is empty (if so, throw error)
348     if (this->message_map[channel].empty()) {
349         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
350         error_str += channel;
351         error_str += " is empty";
352
353         #ifdef _WIN32
354             std::cout << error_str << std::endl;
355         #endif /* _WIN32 */
356
357         throw std::runtime_error(error_str);
358     }
359
360     // 3. pop message
361     this->message_map[channel].pop_front();
362
363     return;
364 } /* 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.

```
284 {
285     // 1. check if channel is in map (if not, throw error)
286     if (this->message_map.count(channel) <= 0) {
287         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
288         error_str += channel;
289         error_str += " is not in message map";
290
291         #ifdef _WIN32
292             std::cout << error_str << std::endl;
293         #endif /* _WIN32 */
294
295         throw std::runtime_error(error_str);
296     }
297
298     // 2. check if channel is empty (if so, throw error)
299     if (this->message_map[channel].empty()) {
300         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
301         error_str += channel;
302         error_str += " is empty";
303
304         #ifdef _WIN32
305             std::cout << error_str << std::endl;
306         #endif /* _WIN32 */
307
308         throw std::runtime_error(error_str);
309     }
310
311     // 3. receive message
312     Message message = this->message_map[channel].front();
313
314     return message;
315 } /* 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.
----------------	--

```
166 {
167     // 1. check if channel is in map (if not, throw error)
168     if (this->message_map.count(channel) <= 0) {
169         std::string error_str = "ERROR MessageHub::removeChannel() channel ";
```

```

170         error_str += channel;
171         error_str += " is not in message map";
172
173         #ifdef _WIN32
174             std::cout << error_str << std::endl;
175         #endif /* _WIN32 */
176         throw std::runtime_error(error_str);
177     }
178
179     // 2. remove channel from map
180     this->message_map[channel].clear();
181     this->message_map.erase(channel);
182
183     std::cout << "Channel " << channel << " removed from message hub" << std::endl;
184
185     return;
186 } /* 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.
----------------	-------------------------

```

205 {
206     // 1. check if channel is in map (if not, throw error)
207     std::string channel = message.channel;
208
209     if (this->message_map.count(channel) <= 0) {
210         std::string error_str = "ERROR MessageHub::sendMessage() channel ";
211         error_str += channel;
212         error_str += " is not in message map";
213
214         #ifdef _WIN32
215             std::cout << error_str << std::endl;
216         #endif /* _WIN32 */
217         throw std::runtime_error(error_str);
218     }
219
220     // 2. send message to message map
221     this->message_map[channel].push_back(message);
222
223     return;
224 } /* 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:

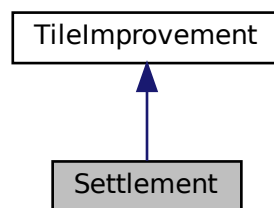
- [header/ESC_core/MessageHub.h](#)
- [source/ESC_core/MessageHub.cpp](#)

4.10 Settlement Class Reference

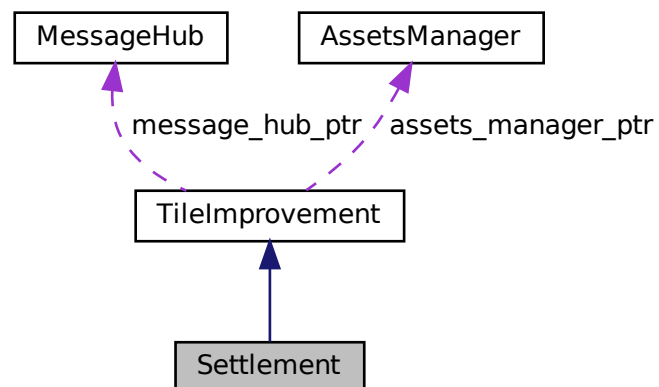
A settlement class (child class of [TileImprovement](#)).

```
#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.

- `std::string` [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- `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

- `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.

```
201 :
202 TileImprovement (
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
210 {
211     // 1. set attributes
212
213     // 1.1. private
214     //...
215
216     // 1.2. public
217     this->tile_improvement_type = TileImprovementType :: SETTLEMENT;
218
219     this->smoke_da = SECONDS_PER_FRAME / 4;
220     this->smoke_dx = 5 * SECONDS_PER_FRAME;
221     this->smoke_dy = -10 * SECONDS_PER_FRAME;
222     this->smoke_prob = 3 * SECONDS_PER_FRAME;
223
224     this->smoke_sprite_list = {};
225
226     this->tile_improvement_string = "SETTLEMENT";
227
228     this->__setUpTileImprovementSpriteStatic();
229
230     std::cout << "Settlement constructed at " << this << std::endl;
231
232     return;
233 } /* Settlement() */
```

4.10.2.2 ~Settlement()

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

Destructor for the [Settlement](#) class.


```

395 {
396     std::cout << "Settlement at " << this << " destroyed" << std::endl;
397
398     return;
399 } /* ~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](#).

```

103 {
104     switch (this->event_ptr->key.code) {
105         //...
106
107         default: {
108             // do nothing!
109
110             break;
111         }
112     }
113
114     return;
115 } /* __handleKeyPressEvents() */

```

4.10.3.2 __handleMouseButtonEvents()

```

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

```

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```

131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139         case (sf::Mouse::Right): {
140             //...
141
142             break;
143         }
144
145         default: {
146             // do nothing!
147
148             break;
149         }
150     }
151
152     return;
153 } /* __handleMouseButtonEvents() */

```

4.10.3.3 __setUpTileImprovementSpriteStatic()

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

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

```
68 {
69     this->tile_improvement_sprite_static.setTexture(
70         *(this->assets_manager_ptr->getTexture("brick_house_64x64_1"))
71     );
72
73     this->tile_improvement_sprite_static.setOrigin(
74         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
75         this->tile_improvement_sprite_static.getLocalBounds().height
76     );
77
78     this->tile_improvement_sprite_static.setPosition(
79         this->position_x,
80         this->position_y - 32
81     );
82
83     this->tile_improvement_sprite_static.setColor(
84         sf::Color(255, 255, 255, 0)
85     );
86
87     return;
88 } /* __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](#).

```
322 {
323     // 1. if just built, call base method and return
324     if (this->just_built) {
325         TileImprovement :: draw();
326
327         return;
328     }
329
330     // 2. draw static sprite and chimney smoke effects
331     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
332
333     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
334
335     double alpha = 255;
336
337     while (iter != this->smoke_sprite_list.end()) {
338         this->render_window_ptr->draw(*iter);
339
340         alpha = (*iter).getColor().a;
341
342         alpha -= this->smoke_da;
343
344         if (alpha <= 0) {
345             iter = this->smoke_sprite_list.erase(iter);
346             continue;
347         }
348
349         (*iter).setColor(sf::Color(255, 255, 255, alpha));
350
351         (*iter).move(
352             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
353             this->smoke_dy
354         );
355
356         (*iter).rotate((((double)rand() / RAND_MAX)));
357
358         iter++;
359     }
```

```

360
361
362     if ((double)rand() / RAND_MAX < smoke_prob) {
363         this->smoke_sprite_list.push_back(
364             sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
365         );
366
367         this->smoke_sprite_list.back().setOrigin(
368             this->smoke_sprite_list.back().getLocalBounds().width / 2,
369             this->smoke_sprite_list.back().getLocalBounds().height / 2
370         );
371
372         this->smoke_sprite_list.back().setPosition(
373             this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
374             this->position_y - 33
375         );
376     }
377
378     this->frame++;
379     return;
380 } /* draw() */

```

4.10.3.5 getTileOptionsSubstring()

```

std::string Settlement::getTileOptionsSubstring (
    void ) [virtual]

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

250 {
251     //          32 char x 17 line console "-----\n";
252     std::string options_substring = "    *** SETTLEMENT OPTIONS *** \n";
253     options_substring += " \n";
254     options_substring += " \n";
255     options_substring += " \n";
256     options_substring += " \n";
257     options_substring += " \n";
258     options_substring += " \n";
259     options_substring += " \n";
260
261     return options_substring;
262 } /* getTileOptionsSubstring() */

```

4.10.3.6 processEvent()

```

void Settlement::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

277 {
278     if (this->event_ptr->type == sf::Event::KeyPressed) {
279         this->__handleKeyPressEvents();
280     }
281
282     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
283         this->__handleMouseButtonEvents();
284     }
285
286     return;
287 } /* processEvent() */

```

4.10.3.7 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
302 {
303     //...
304
305     return;
306 } /* processMessage() */
```

4.10.4 Member Data Documentation

4.10.4.1 smoke_da

```
double Settlement::smoke_da
```

The per frame delta in smoke particle alpha value.

4.10.4.2 smoke_dx

```
double Settlement::smoke_dx
```

The per frame delta in smoke particle x position.

4.10.4.3 smoke_dy

```
double Settlement::smoke_dy
```

The per frame delta in smoke particle y position.

4.10.4.4 smoke_prob

```
double Settlement::smoke_prob
```

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

4.10.4.5 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:

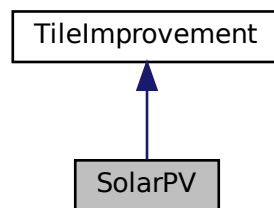
- header/[Settlement.h](#)
- source/[Settlement.cpp](#)

4.11 SolarPV Class Reference

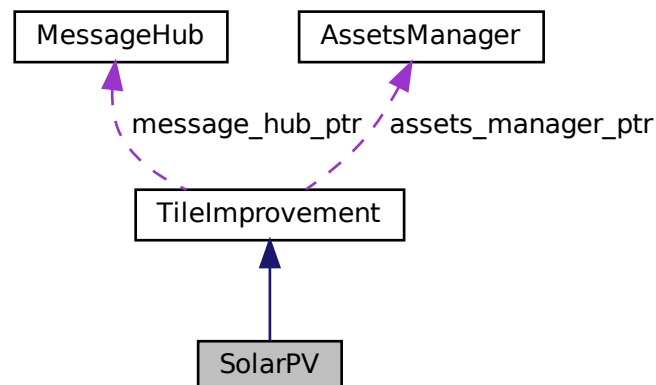
A settlement class (child class of [TileImprovement](#)).

```
#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.
- std::string [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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.

```

201 :
202 TileImprovement (
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
210 {
211     // 1. set attributes
212
213     // 1.1. private
214     //...
215
216     // 1.2. public
217     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
218
219     this->is_running = false;
220
221     this->health = 100;
222
223     this->tile_improvement_string = "SOLAR PV ARRAY";
224
225     this->__setUpTileImprovementSpriteStatic();
226
227     std::cout << "SolarPV constructed at " << this << std::endl;
228
229     return;
230 } /* SolarPV() */

```

4.11.2.2 ~SolarPV()

```

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

```

Destructor for the [SolarPV](#) class.

```

352 {
353     std::cout << "SolarPV at " << this << " destroyed" << std::endl;
354
355     return;
356 } /* ~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](#).

```
103 {
104     switch (this->event_ptr->key.code) {
105         //...
106
107         default: {
108             // do nothing!
109
110             break;
111         }
112     }
113 }
114
115 return;
116 } /* __handleKeyPressEvents() */
```

4.11.3.2 `__handleMouseButtonEvents()`

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

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139         case (sf::Mouse::Right): {
140             //...
141
142             break;
143         }
144
145         default: {
146             // do nothing!
147
148             break;
149         }
150     }
151 }
152
153 return;
154 } /* __handleMouseButtonEvents() */
```


4.11.3.3 __setUpTileImprovementSpriteStatic()

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

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

```
68 {
69     this->tile_improvement_sprite_static.setTexture(
70         *(this->assets_manager_ptr->getTexture("solar PV array"))
71     );
72     this->tile_improvement_sprite_static.setOrigin(
73         this->tile_improvement_sprite_static.getLocalBounds().width / 2,
74         this->tile_improvement_sprite_static.getLocalBounds().height
75     );
76     this->tile_improvement_sprite_static.setPosition(
77         this->position_x,
78         this->position_y - 32
79     );
80     this->tile_improvement_sprite_static.setColor(
81         sf::Color(255, 255, 255, 0)
82     );
83     return;
84 } /* __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](#).

```
323 {
324     // 1. if just built, call base method and return
325     if (this->just_built) {
326         TileImprovement::draw();
327     }
328     return;
329 }
330
331 // 1. draw static sprite
332 this->render_window_ptr->draw(this->tile_improvement_sprite_static);
333
334 this->frame++;
335 return;
336 } /* draw() */
```

4.11.3.5 getTileOptionsSubstring()

```
std::string SolarPV::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

247 {
248     //          32 char x 17 line console "-----\n";
249     std::string options_substring = "    **** SOLAR PV OPTIONS **** \n";
250     options_substring += " \n";
251     options_substring += " \n";
252     options_substring += " \n";
253     options_substring += " \n";
254     options_substring += " \n";
255     options_substring += " \n";
256     options_substring += " \n";
257
258     options_substring += "[P]:  SCRAP (";
259     options_substring += std::to_string(SCRAP_COST);
260     options_substring += " K)";
261
262     return options_substring;
263 } /* getTileOptionsSubstring() */

```

4.11.3.6 processEvent()

```

void SolarPV::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

278 {
279     if (this->event_ptr->type == sf::Event::KeyPressed) {
280         this->__handleKeyPressEvents();
281     }
282
283     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
284         this->__handleMouseButtonEvents();
285     }
286
287     return;
288 } /* processEvent() */

```

4.11.3.7 processMessage()

```

void SolarPV::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

303 {
304     //...
305
306     return;
307 } /* processMessage() */

```

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

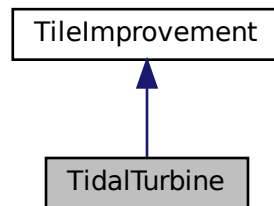
- header/[SolarPV.h](#)
- source/[SolarPV.cpp](#)

4.12 TidalTurbine Class Reference

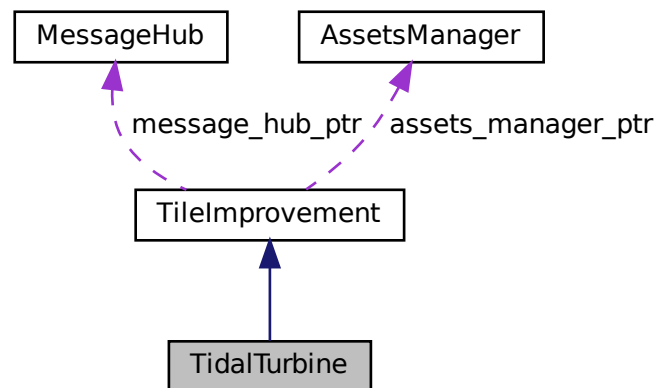
A settlement class (child class of [TileImprovement](#)).

```
#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.
- std::string [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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.

```

212 :
213 TileImprovement (
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
221 {
222     // 1. set attributes
223
224     // 1.1. private
225     //...
226
227     // 1.2. public
228     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
229
230     this->is_running = false;
231
232     this->tile_improvement_string = "TIDAL TURBINE";
233
234     this->__setUpTileImprovementSpriteAnimated();
235
236     std::cout << "TidalTurbine constructed at " << this << std::endl;
237
238     return;
239 } /* TidalTurbine() */

```

4.12.2.2 ~TidalTurbine()

```

TidalTurbine::~TidalTurbine (
    void ) [virtual]

```

Destructor for the [TidalTurbine](#) class.

```

373 {
374     std::cout << "TidalTurbine at " << this << " destroyed" << std::endl;
375
376     return;
377 } /* ~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](#).

```

114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118         default: {
119             // do nothing!
120
121             break;
122         }
123     }
124
125     return;
126 } /* __handleKeyPressEvents() */

```

4.12.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146             break;
147         }
148     }
149
150     case (sf::Mouse::Right): {
151         //...
152         break;
153     }
154
155     default: {
156         // do nothing!
157         break;
158     }
159 }
160
161 return;
162 } /* __handleMouseButtonEvents() */
```

4.12.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet (
70         *(this->assets_manager_ptr->getTexture("tidal turbine"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("tidal turbine")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 } /* __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](#).

```
332 {
333     // 1. if just built, call base method and return
334     if (this->just_built) {
335         TileImprovement :: draw();
336     }
337     return;
338 }
339
340
341 // 1. draw first element of animated sprite
342 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
343
344
345 // 2. draw second element of animated sprite
346 if (this->is_running) {
347     //...
348 }
349
350 else {
351     //...
352 }
353
354 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
355
356 this->frame++;
357 return;
358 } /* draw() */
```

4.12.3.5 getTileOptionsSubstring()

```
std::string TidalTurbine::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
256 {
257     // 32 char x 17 line console "-----\n";
258     std::string options_substring = "**** TIDAL TURBINE OPTIONS **** \n";
259     options_substring += " \n";
260     options_substring += " \n";
261     options_substring += " \n";
262     options_substring += " \n";
263     options_substring += " \n";
264     options_substring += " \n";
265     options_substring += " \n";
266
267     options_substring += "[P]: SCRAP (";
268     options_substring += std::to_string(SCRAP_COST);
269     options_substring += " K)";
270
271     return options_substring;
272 } /* getTileOptionsSubstring() */
```

4.12.3.6 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
287 {
288     if (this->event_ptr->type == sf::Event::KeyPressed) {
289         this->__handleKeyPressEvents();
290     }
291
292     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
293         this->__handleMouseButtonEvents();
294     }
295
296     return;
297 } /* processEvent() */
```

4.12.3.7 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
312 {
313     //...
314
315     return;
316 } /* processMessage() */
```

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

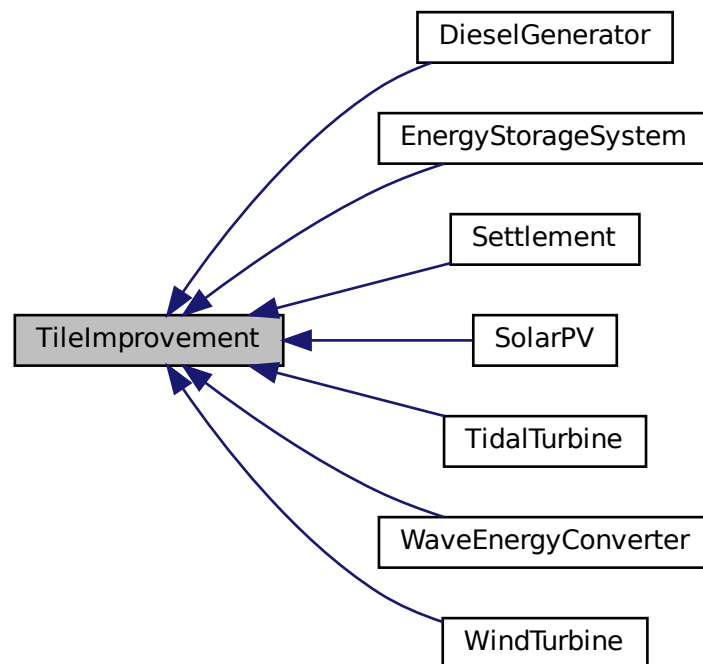
- header/[TidalTurbine.h](#)
- source/[TidalTurbine.cpp](#)

4.13 TileImprovement Class Reference

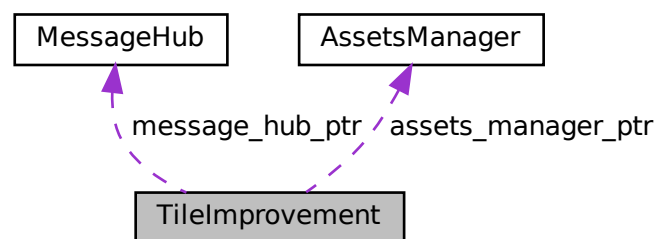
A base class for the tile improvement hierarchy.

```
#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.
- void [setIsSelected](#) (bool)
Method to set the is selected attribute.

- virtual std::string [getTileOptionsSubstring](#) (void)
- 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.
- bool [production_menu_open](#)
A boolean which indicates whether or not the production menu is open.
- unsigned long long int [frame](#)
The current frame of this object.
- int [credits](#)
The current balance of credits.
- int [health](#)
The health of the improvement.
- 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.
- sf::RectangleShape [production_menu_backing](#)
A backing for the production build menu.
- sf::Text [production_menu_backing_text](#)
Text for the production menu backing.

Protected Member Functions

- void [__setUpProductionMenu](#) (void)
Helper method to set up and position production menu assets (drawable).
- virtual void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- virtual void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__openProductionMenu](#) (void)
Helper method to open the production menu.
- void [__closeProductionMenu](#) (void)
Helper method to close the production menu.

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.

```

255 {
256     // 1. set attributes
257
258     // 1.1. protected
259     this->event_ptr = event_ptr;
260     this->render_window_ptr = render_window_ptr;
261
262     this->assets_manager_ptr = assets_manager_ptr;
263     this->message_hub_ptr = message_hub_ptr;
264
265     // 1.2. public
266     this->is_selected = true;
267     this->just_built = true;
268     this->production_menu_open = false;
269
270     this->frame = 0;
271     this->credits = 0;
272
273     this->position_x = position_x;
274     this->position_y = position_y;
275
276     this->game_phase = "build settlement";
277
278     this->__setUpProductionMenu();
279
280     std::cout << "TileImprovement constructed at " << this << std::endl;
281
282     return;
283 } /* TileImprovement() */

```

4.13.2.2 ~TileImprovement()

```

TileImprovement::~TileImprovement (
    void ) [virtual]

```

Destructor for the [TileImprovement](#) class.

```

515 {
516     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
517
518     return;
519 } /* ~TileImprovement() */

```

4.13.3 Member Function Documentation

4.13.3.1 __closeProductionMenu()

```

void TileImprovement::__closeProductionMenu (
    void ) [protected]

```

Helper method to close the production menu.

```

199 {
200     if (not this->production_menu_open) {
201         return;
202     }
203
204     this->production_menu_open = false;
205     this->assets_manager_ptr->getSound("build menu close")->play();
206
207     return;
208 } /* __closeProductionMenu() */

```

4.13.3.2 __handleKeyPressEvents()

```

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

```

Helper method to handle key press events.

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

```

104 {
105     switch (this->event_ptr->key.code) {
106         case (sf::Keyboard::E): {
107             this->__openProductionMenu();
108
109             break;
110         }
111
112         default: {
113             // do nothing!
114
115             break;
116         }
117     }
118 }
119
120 return;
121 } /* __handleKeyPressEvents() */

```

4.13.3.3 __handleMouseButtonEvents()

```

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

```

Helper method to handle mouse button events.

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

```

136 {
137     switch (this->event_ptr->mouseButton.button) {
138         case (sf::Mouse::Left): {
139             //...
140
141             break;
142         }
143
144         case (sf::Mouse::Right): {
145             //...
146
147             break;
148         }
149     }
150
151     default: {
152         // do nothing!
153
154         break;
155     }
156 }
157
158 return;
159 } /* __handleMouseButtonEvents() */

```

4.13.3.4 __openProductionMenu()

```
void TileImprovement::__openProductionMenu (
    void ) [protected]
```

Helper method to open the production menu.

```
175 {
176     if (this->production_menu_open) {
177         return;
178     }
179
180     this->production_menu_open = true;
181     this->assets_manager_ptr->getSound("build menu open")->play();
182
183     return;
184 } /* __openProductionMenu() */
```

4.13.3.5 __setUpProductionMenu()

```
void TileImprovement::__setUpProductionMenu (
    void ) [protected]
```

Helper method to set up and position production menu assets (drawable).

```
68 {
69     // 1. set up and place build menu backing and text
70     this->production_menu_backing.setSize(sf::Vector2f(400, 256));
71     this->production_menu_backing.setOrigin(200, 128);
72     this->production_menu_backing.setPosition(400, 400);
73     this->production_menu_backing.setFillColor(MONOCROME_SCREEN_BACKGROUND);
74     this->production_menu_backing.setOutlineColor(MENU_FRAME_GREY);
75     this->production_menu_backing.setOutlineThickness(4);
76
77     this->production_menu_backing_text.setString("**** PRODUCTION MENU ****");
78     this->production_menu_backing_text.setFont(
79         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
80     );
81     this->production_menu_backing_text.setCharacterSize(16);
82     this->production_menu_backing_text.setFillColor(MONOCROME_TEXT_GREEN);
83     this->production_menu_backing_text.setOrigin(
84         this->production_menu_backing_text.getLocalBounds().width / 2, 0
85     );
86     this->production_menu_backing_text.setPosition(200, 200 - 128 + 4);
87
88     return;
89 } /* __setUpProductionMenu() */
```

4.13.3.6 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](#).

```
386 {
387     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
388         int alpha = this->tile_improvement_sprite_static.getColor().a;
389
390         alpha += 0.08 * FRAMES_PER_SECOND;
391
392         this->tile_improvement_sprite_static.setColor(
393             sf::Color(255, 255, 255, alpha)
394         );
395     }
```

```

395
396     this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);
397
398     if (
399         (alpha >= 255) or
400         (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
401     ) {
402         this->tile_improvement_sprite_static.setColor(
403             sf::Color(255, 255, 255, 255)
404         );
405
406         this->tile_improvement_sprite_static.setPosition(
407             this->position_x,
408             this->position_y + 12
409         );
410
411         this->just_built = false;
412         this->assets_manager_ptr->getSound("place_improvement")->play();
413     }
414
415     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
416 }
417
418
419 else {
420     int alpha = 0;
421
422     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
423         alpha = this->tile_improvement_sprite_animated[i].getColor().a;
424
425         alpha += 0.08 * FRAMES_PER_SECOND;
426
427         this->tile_improvement_sprite_animated[i].setColor(
428             sf::Color(255, 255, 255, alpha)
429         );
430
431         this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
432
433         if (
434             (alpha >= 255) or
435             (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
436         ) {
437             this->tile_improvement_sprite_animated[i].setColor(
438                 sf::Color(255, 255, 255, 255)
439             );
440
441             this->tile_improvement_sprite_animated[i].setPosition(
442                 this->position_x,
443                 this->position_y + 12
444             );
445         }
446
447         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
448     }
449
450     if (
451         (alpha >= 255) or
452         (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
453     ) {
454         this->just_built = false;
455         this->assets_manager_ptr->getSound("place_improvement")->play();
456
457         switch (this->tile_improvement_type) {
458             case (TileImprovementType :: WIND_TURBINE): {
459                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
460                     this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
461                     this->tile_improvement_sprite_animated[i].move(0, -32);
462                 }
463
464                 break;
465             }
466
467             case (TileImprovementType :: TIDAL_TURBINE): {
468                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
469                     this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
470                     this->tile_improvement_sprite_animated[i].move(0, -19);
471                 }
472
473                 break;
474             }
475
476             case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
477                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
478                     this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
479                     this->tile_improvement_sprite_animated[i].move(0, -32);
480                 }
481

```

```

482         }
483
484         break;
485     }
486
487     default: {
488         // do nothing!
489
490         break;
491     }
492 }
493
494 }
495 }
496
497
498     this->frame++;
499     return;
500 } /* draw() */

```

4.13.3.7 getTitleOptionsSubstring()

```

virtual std::string TileImprovement::getTitleOptionsSubstring (
    void ) [inline], [virtual]

```

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

```

143 {return "";}

```

4.13.3.8 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](#).

```

341 {
342     if (this->event_ptr->type == sf::Event::KeyPressed) {
343         this->__handleKeyPressEvents();
344     }
345
346     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
347         this->__handleMouseButtonEvents();
348     }
349
350     return;
351 } /* processEvent() */

```

4.13.3.9 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](#).

```

366 {
367     //...
368
369     return;
370 } /* processMessage() */

```


4.13.3.10 setIsSelected()

```
void TileImprovement::setIsSelected (
    bool is_selected )
```

Method to set the is selected attribute.

Parameters

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

```
300 {
301     this->is_selected = is_selected;
302
303     if (is_selected) {
304         switch (this->tile_improvement_type) {
305             /*
306              * case (TileImprovementType :: SETTLEMENT): {
307                  this->assets_manager_ptr->getSound("people and children")->play();
308              }
309              break;
310             */
311             /*
312              * default: {
313                  // do nothing!
314              }
315              break;
316             */
317         }
318     }
319 }
320
321 if ((not is_selected) and this->production_menu_open) {
322     this->__closeProductionMenu();
323 }
324
325 return;
326 } /* setIsSelected() */
```

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

```
unsigned long long 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 health

```
int TileImprovement::health
```

The health of the improvement.

4.13.4.7 is_running

```
bool TileImprovement::is_running
```

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

4.13.4.8 is_selected

```
bool TileImprovement::is_selected
```

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

4.13.4.9 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.10 message_hub_ptr

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

A pointer to the message hub.

4.13.4.11 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.12 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.13 production_menu_backing

```
sf::RectangleShape TileImprovement::production_menu_backing
```

A backing for the production build menu.

4.13.4.14 production_menu_backing_text

```
sf::Text TileImprovement::production_menu_backing_text
```

Text for the production menu backing.

4.13.4.15 production_menu_open

```
bool TileImprovement::production_menu_open
```

A boolean which indicates whether or not the production menu is open.

4.13.4.16 render_window_ptr

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

A pointer to the render window.

4.13.4.17 tile_improvement_sprite_animated

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

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

4.13.4.18 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.19 tile_improvement_string

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

A string representation of the tile improvement type.

4.13.4.20 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:

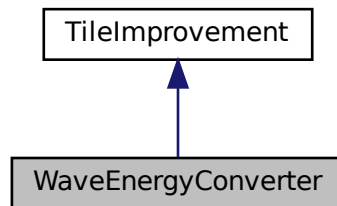
- header/[TileImprovement.h](#)
- source/[TileImprovement.cpp](#)

4.14 WaveEnergyConverter Class Reference

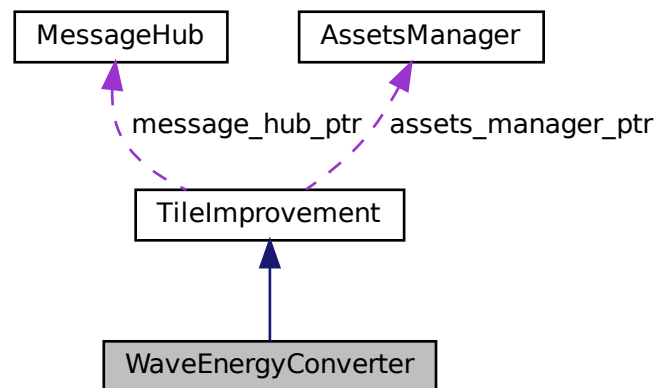
A settlement class (child class of [TileImprovement](#)).

```
#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.
- std::string [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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.

```

212 :
213 TileImprovement (
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
221 {
222     // 1. set attributes
223
224     // 1.1. private
225     //...
226
227     // 1.2. public
228     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
229
230     this->is_running = false;
231
232     this->health = 100;
233
234     this->tile_improvement_string = "WAVE ENERGY";
235
236     this->__setUpTileImprovementSpriteAnimated();
237
238     std::cout << "WaveEnergyConverter constructed at " << this << std::endl;
239
240     return;
241 } /* WaveEnergyConverter() */

```

4.14.2.2 ~WaveEnergyConverter()

```

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

```

Destructor for the [WaveEnergyConverter](#) class.

```

375 {
376     std::cout << "WaveEnergyConverter at " << this << " destroyed" << std::endl;
377
378     return;
379 } /* ~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](#).

```

114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118         default: {
119             // do nothing!
120
121             break;
122         }
123     }
124
125     return;
126 } /* __handleKeyPressEvents() */

```

4.14.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146             break;
147         }
148     }
149
150     case (sf::Mouse::Right): {
151         //...
152         break;
153     }
154
155     default: {
156         // do nothing!
157         break;
158     }
159 }
160
161 return;
162 } /* __handleMouseButtonEvents() */
```

4.14.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet (
70         *(this->assets_manager_ptr->getTexture("wave energy converter"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("wave energy converter")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 } /* __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](#).

```
334 {
335     // 1. if just built, call base method and return
336     if (this->just_built) {
337         TileImprovement :: draw();
338     }
339     return;
340 }
341
342
343 // 1. draw first element of animated sprite
344 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
345
346
347 // 2. draw second element of animated sprite
348 if (this->is_running) {
349     //...
350 }
351
352 else {
353     //...
354 }
355
356 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
357
358 this->frame++;
359 return;
360 } /* draw() */
```

4.14.3.5 getTileOptionsSubstring()

```
std::string WaveEnergyConverter::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
258 {
259     // 32 char x 17 line console "-----\n";
260     std::string options_substring = " **** WAVE ENERGY OPTIONS **** \n";
261     options_substring += " \n";
262     options_substring += " \n";
263     options_substring += " \n";
264     options_substring += " \n";
265     options_substring += " \n";
266     options_substring += " \n";
267     options_substring += " \n";
268
269     options_substring += "[P]: SCRAP (";
270     options_substring += std::to_string(SCRAP_COST);
271     options_substring += " K)";
272
273     return options_substring;
274 } /* getTileOptionsSubstring() */
```

4.14.3.6 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
289 {
290     if (this->event_ptr->type == sf::Event::KeyPressed) {
291         this->__handleKeyPressEvents();
292     }
293
294     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
295         this->__handleMouseButtonEvents();
296     }
297
298     return;
299 } /* processEvent() */
```

4.14.3.7 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
314 {
315     //...
316
317     return;
318 } /* processMessage() */
```

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

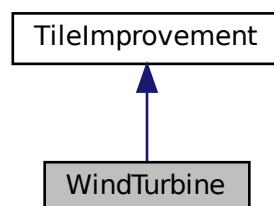
- header/[WaveEnergyConverter.h](#)
- source/[WaveEnergyConverter.cpp](#)

4.15 WindTurbine Class Reference

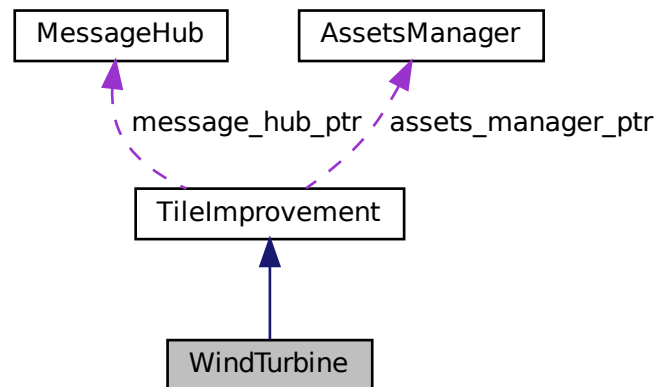
A settlement class (child class of [TileImprovement](#)).

```
#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.
- std::string [getTileOptionsSubstring](#) (void)
Helper method to assemble and return tile options substring.
- 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.

```
212 :
213 TileImprovement (
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
221 {
222     // 1. set attributes
223
224     // 1.1. private
225     //...
226
227     // 1.2. public
228     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
229
230     this->is_running = false;
231
232     this->health = 100;
233
234     this->tile_improvement_string = "WIND TURBINE";
235
236     this->__setUpTileImprovementSpriteAnimated();
237
238     std::cout << "WindTurbine constructed at " << this << std::endl;
239
240     return;
241 } /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

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

Destructor for the [WindTurbine](#) class.

```
375 {  
376     std::cout << "WindTurbine at " << this << " destroyed" << std::endl;  
377  
378     return;  
379 } /* ~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](#).

```
114 {  
115     switch (this->event_ptr->key.code) {  
116         //...  
117  
118         default: {  
119             // do nothing!  
120  
121             break;  
122         }  
123     }  
124 }  
125  
126 return;  
127 } /* __handleKeyPressEvents() */
```

4.15.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from [TileImprovement](#).

```
142 {  
143     switch (this->event_ptr->mouseButton.button) {  
144         case (sf::Mouse::Left): {  
145             //...  
146  
147             break;  
148         }  
149  
150         case (sf::Mouse::Right): {  
151             //...  
152  
153             break;  
154         }  
155  
156         default: {  
157             // do nothing!  
158  
159             break;  
160         }  
161     }  
162 }  
163 }  
164  
165 return;  
166 } /* __handleMouseButtonEvents() */
```

4.15.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet (
70         *(this->assets_manager_ptr->getTexture("wind turbine"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("wind turbine")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 } /* __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](#).

```
334 {
335     // 1. if just built, call base method and return
336     if (this->just_built) {
337         TileImprovement::draw();
338     }
339     return;
340 }
341
342
343 // 1. draw first element of animated sprite
344 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
345
346
347 // 2. draw second element of animated sprite
348 if (this->is_running) {
349     //...
350 }
351
352 else {
353     //...
354 }
355
356 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
357
358 this->frame++;
359 return;
360 } /* draw() */
```

4.15.3.5 getTileOptionsSubstring()

```
std::string WindTurbine::getTileOptionsSubstring (
    void ) [virtual]
```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
258 {
259     //          32 char x 17 line console "-----\n";
260     std::string options_substring          = " **** WIND TURBINE OPTIONS **** \n";
261     options_substring                      += " \n";
262     options_substring                      += " \n";
263     options_substring                      += " \n";
264     options_substring                      += " \n";
265     options_substring                      += " \n";
266     options_substring                      += " \n";
267     options_substring                      += " \n";
268
269     options_substring                      += "[P]: SCRAP (";
270     options_substring                      += std::to_string(SCRAP_COST);
271     options_substring                      += " K)";
272
273     return options_substring;
274 } /* getTileOptionsSubstring() */
```

4.15.3.6 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
289 {
290     if (this->event_ptr->type == sf::Event::KeyPressed) {
291         this->__handleKeyPressEvents();
292     }
293
294     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
295         this->__handleMouseButtonEvents();
296     }
297
298     return;
299 } /* processEvent() */
```

4.15.3.7 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
314 {
315     //...
316
317     return;
318 } /* processMessage() */
```

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

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

File Documentation

Header file for the `ContextMenu` class.

```

graph TD
    A[ImageNet-1000] --> B[ImageNet-1000-1]
    A --> C[ImageNet-1000-2]
    A --> D[ImageNet-1000-3]
    B --> E[ImageNet-1000-4]
    C --> E
    D --> E
    E --> F[airplane]
    E --> G[bird]
    E --> H[cat]
    E --> I[deer]
    E --> J[dog]
    E --> K[elephant]
    E --> L[giraffe]
    E --> M[horse]
    E --> N[lion]
    E --> O[monkey]
    E --> P[sheep]
    E --> Q[tiger]
    E --> R[truck]
    E --> S[whale]
    E --> T[zebra]
    E --> U[other categories]
  
```

```
graph BT; GameCpp[source/Game.cpp] --> GameH[header/Game.h]; MainCpp[source/main.cpp] --> GameH; GameH --> ContextMenuH[header/ContextMenu.h]; ContextMenuCpp[source/ContextMenu.cpp] --> ContextMenuH;
```

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.

```

68     {
69         NONE_STATE,
70         READY,
71         MENU,
72         TILE,
73         N_CONSOLE_STATES
74     };

```

5.2 header/DieselGenerator.h File Reference

Header file for the [DieselGenerator](#) class.

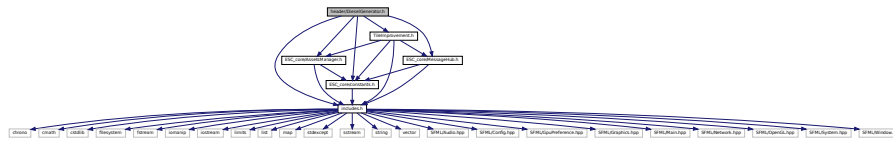
```

#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"

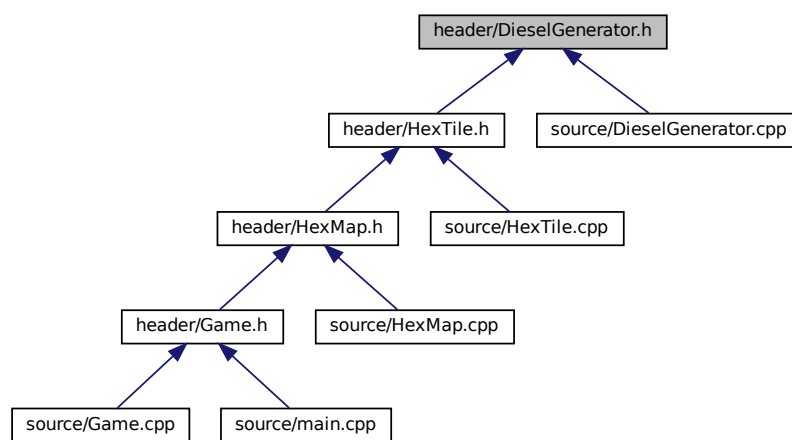
```

```
#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.

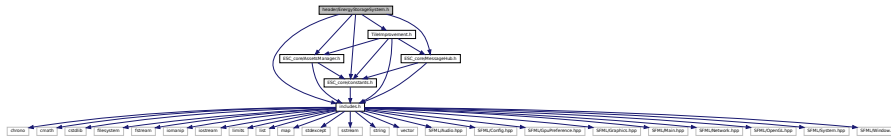
5.3 header/EnergyStorageSystem.h File Reference

Header file for the `EnergyStorageSystem` class.

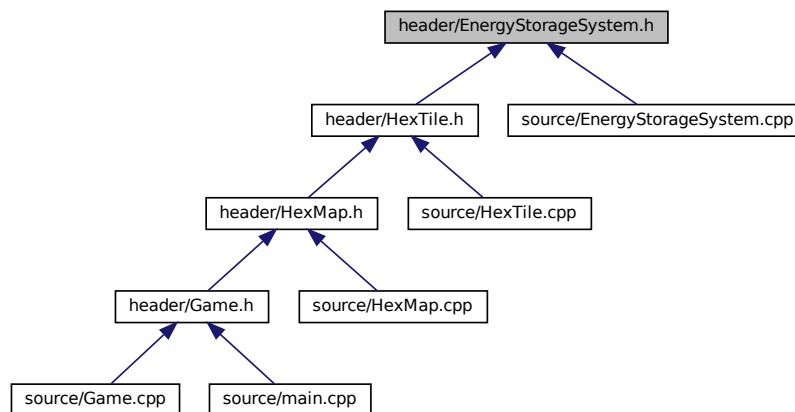
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

```
#include "TileImprovement.h"
```

Include dependency graph for EnergyStorageSystem.h:



This graph shows which files directly or indirectly include this file:



Classes

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

5.3.1 Detailed Description

Header file for the [EnergyStorageSystem](#) class.

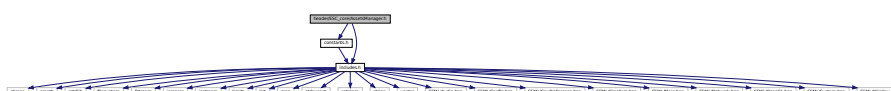
5.4 header/ESC_core/AssetsManager.h File Reference

Header file for the [AssetsManager](#) class.

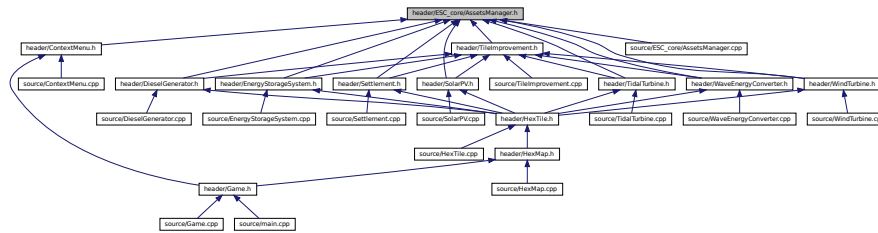
```
#include "constants.h"
```

```
#include "includes.h"
```

Include dependency graph for AssetsManager.h:



This graph shows which files directly or indirectly include this file:



Classes

- class `AssetsManager`
A class which manages visual and sound assets.

5.4.1 Detailed Description

Header file for the `AssetsManager` class.

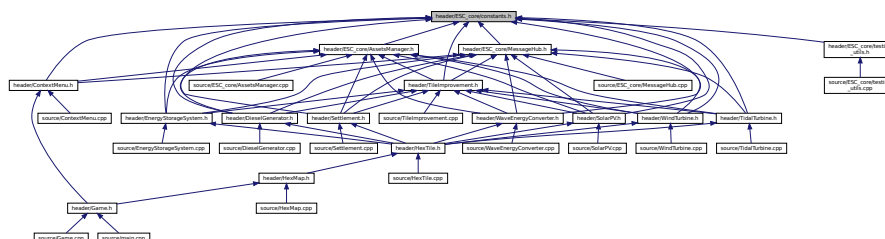
5.5 header/ESC_core/constants.h File Reference

Header file for various constants.

```
#include "includes.h"
Include dependency graph for constants.h:
```



This graph shows which files directly or indirectly include this file:



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 `SCRAP_COST` = 50
- The cost of scrapping a tile improvement (other than settlement).*
- 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 SCRAP_COST

```
const int SCRAP_COST = 50
```

The cost of scrapping a tile improvement (other than settlement).

5.5.3.19 SECONDS_PER_FRAME

```
const double SECONDS_PER_FRAME = 1.0 / 60
```

Target seconds per frame (just reciprocal of target frames per second).

5.5.3.20 SECONDS_PER_MONTH

```
const unsigned long long int SECONDS_PER_MONTH = 2628164
```

5.5.3.21 SECONDS_PER_YEAR

```
const unsigned long long int SECONDS_PER_YEAR = 31537970
```

5.5.3.22 SOLAR_PV_BUILD_COST

```
const int SOLAR_PV_BUILD_COST = 300
```

The cost of building (or upgrading) a solar PV array.

5.5.3.23 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.24 STARTING_CREDITS

```
const int STARTING_CREDITS = 99999
```

The starting balance of credits.

5.5.3.25 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.26 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

The cost of building (or upgrading) a tidal turbine.

5.5.3.27 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).

5.5.3.28 TILE_SELECTED_CHANNEL

```
const std::string TILE_SELECTED_CHANNEL = "TILE SELECTED CHANNEL"
```

A message channel for tile selection messages.

5.5.3.29 TILE_STATE_CHANNEL

```
const std::string TILE_STATE_CHANNEL = "TILE STATE CHANNEL"
```

A message channel for tile state messages.

5.5.3.30 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).

5.5.3.31 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800
```

The cost of building (or upgrading) a wave energy converter.

5.5.3.32 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

The cost of building (or upgrading) a wind turbine.

5.5.3.33 WIND_TURBINE_WATER_BUILD_MULTIPLIER

```
const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25
```

The additional cost of building on water.

5.6 header/ESC_core/doxygen_cite.h File Reference

Header file which simply cites the doxygen tool.

5.6.1 Detailed Description

Header file which simply cites the doxygen tool.

Ref: [van Heesch. \[2023\]](#)

5.7 header/ESC_core/includes.h File Reference

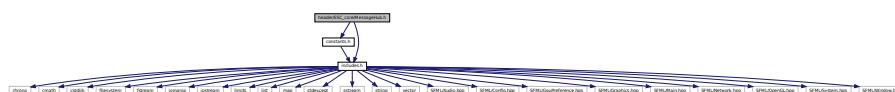
Header file for various includes.

```
#include <chrono>
#include <cmath>
#include <cstdlib>
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <limits>
#include <list>
#include <map>
#include <stdexcept>
#include <sstream>
#include <string>
#include <vector>
#include <SFML/Audio.hpp>
#include <SFML/Config.hpp>
#include <SFML/GpuPreference.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Main.hpp>
#include <SFML/Network.hpp>
```


Figure 1 illustrates the structure of the 'Networks' dataset. The dataset is composed of 18 different network types, each represented by a box. The 'Networks' dataset is shown at the top, with arrows pointing down to each of the 18 network types. The network types are: chao, creat, cdb, floccnet, fennel, giranet, isbnet, isdc, isf, map, cbaocd, isbnet, isg, isct, SML_hada, SML_Catlg, SML_Equivalence, SML_Graphs, SML_Mat, SML_Network, SML_Signed, SML_Signed, and SML_Signed.

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

```
#include "constants.h"
#include "includes.h"
Include dependency graph for MessageHub.h:
```

[illegible]

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 " <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> ").

```
462 {
463     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
464     error_str += std::to_string(line);
```

```

465     error_str += " of ";
466     error_str += file;
467
468     #ifdef _WIN32
469         std::cout << error_str << std::endl;
470     #endif
471
472     throw std::runtime_error(error_str);
473     return;
474 } /* 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.
------------------	---

```

114 {
115     std::cout << "\x1B[33m" << input_str << "\033[0m";
116     return;
117 } /* 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.
------------------	---

```

94 {
95     std::cout << "\x1B[32m" << input_str << "\033[0m";
96     return;
97 } /* 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 <code>std::cout</code> .
------------------	---

```

134 {
135     std::cout << "\x1B[31m" << input_str << "\033[0m";
136     return;
137 } /* 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> ").

```

168 {
169     if (fabs(x - y) <= FLOAT_TOLERANCE) {
170         return;
171     }
172
173     std::string error_str = "ERROR: testFloatEquals():\t in ";
174     error_str += file;
175     error_str += "\tline ";
176     error_str += std::to_string(line);
177     error_str += ":\t\n";
178     error_str += std::to_string(x);
179     error_str += " and ";
180     error_str += std::to_string(y);
181     error_str += " are not equal to within +/- ";
182     error_str += std::to_string(FLOAT_TOLERANCE);
183     error_str += "\n";
184
185     #ifdef WIN32
186         std::cout << error_str << std::endl;
187     #endif
188
189     throw std::runtime_error(error_str);
190     return;
191 } /* 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__").

```

221 {
222     if (x > y) {
223         return;
224     }
225
226     std::string error_str = "ERROR: testGreaterThan():\t in ";
227     error_str += file;
228     error_str += "\tline ";
229     error_str += std::to_string(line);
230     error_str += ":\t\n";
231     error_str += std::to_string(x);
232     error_str += " is not greater than ";
233     error_str += std::to_string(y);
234     error_str += "\n";
235
236     #ifdef _WIN32
237         std::cout << error_str << std::endl;
238     #endif
239
240     throw std::runtime_error(error_str);
241     return;
242 } /* 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__").

```

272 {
273     if (x >= y) {
274         return;
275     }
276
277     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
278     error_str += file;
279     error_str += "\tline ";
280     error_str += std::to_string(line);
281     error_str += ":\t\n";
282     error_str += std::to_string(x);
283     error_str += " is not greater than or equal to ";
284     error_str += std::to_string(y);
285     error_str += "\n";
286
287     #ifdef _WIN32
288         std::cout << error_str << std::endl;
289     #endif
290
291     throw std::runtime_error(error_str);

```

```

292     return;
293 } /* 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__").

```

323 {
324     if (x < y) {
325         return;
326     }
327
328     std::string error_str = "ERROR: testLessThan():\t in ";
329     error_str += file;
330     error_str += "\tline ";
331     error_str += std::to_string(line);
332     error_str += ":\t\n";
333     error_str += std::to_string(x);
334     error_str += " is not less than ";
335     error_str += std::to_string(y);
336     error_str += "\n";
337
338     #ifdef _WIN32
339         std::cout << error_str << std::endl;
340     #endif
341
342     throw std::runtime_error(error_str);
343     return;
344 } /* 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__").

```

374 {
375     if (x <= y) {
376         return;
377     }
378
379     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
380     error_str += file;
381     error_str += "\tline ";
382     error_str += std::to_string(line);
383     error_str += ":\t\n";
384     error_str += std::to_string(x);
385     error_str += " is not less than or equal to ";
386     error_str += std::to_string(y);
387     error_str += "\n";
388
389     #ifdef _WIN32
390         std::cout << error_str << std::endl;
391     #endif
392
393     throw std::runtime_error(error_str);
394     return;
395 } /* 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__").

```

422 {
423     if (statement) {
424         return;
425     }
426
427     std::string error_str = "ERROR: testTruth():\t in ";
428     error_str += file;
429     error_str += "\tline ";
430     error_str += std::to_string(line);
431     error_str += ":\t\n";
432     error_str += "Given statement is not true";
433
434     #ifdef _WIN32
435         std::cout << error_str << std::endl;
436     #endif
437
438     throw std::runtime_error(error_str);
439     return;
440 } /* testTruth() */

```

5.10 header/Game.h File Reference

```

#include "HexMap.h"
#include "ContextMenu.h"

```


Classes

- class [HexMap](#)

A class which defines a hex map of hex tiles.

5.11.1 Detailed Description

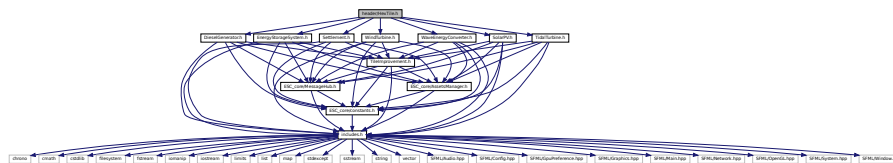
Header file for the [HexMap](#) class.

5.12 header/HexTile.h File Reference

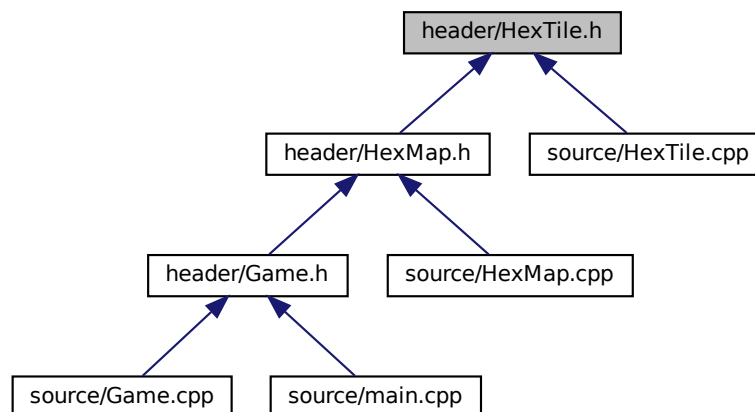
Header file for the [Game](#) class.

```
#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.

```

88         {
89     POOR,
90     BELOW_AVERAGE,
91     AVERAGE,
92     ABOVE_AVERAGE,
93     GOOD,
94     N_TILE_RESOURCES
95 };  /* 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.

```

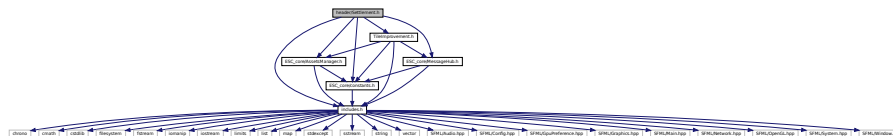
71         {
72             NONE_TYPE,
73             FOREST,
74             LAKE,
75             MOUNTAINS,
76             OCEAN,
77             PLAINS,
78             N_TILE_TYPES
79 }; /* TileType */

```

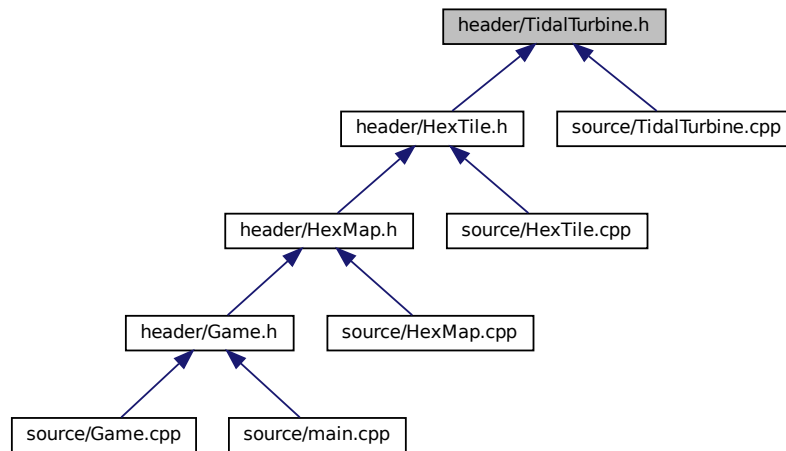
5.13 header/Settlement.h File Reference

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

5.15.1 Detailed Description

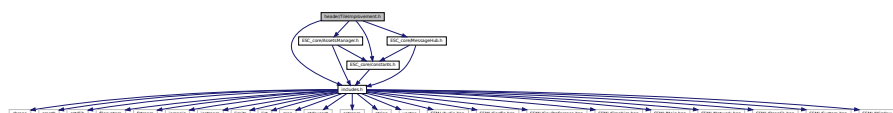
Header file for the [TidalTurbine](#) class.

5.16 header/TileImprovement.h File Reference

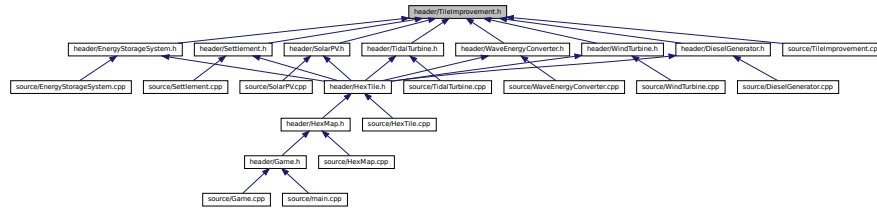
Header file for the [TileImprovement](#) class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

Include dependency graph for `TileImprovement.h`:



This graph shows which files directly or indirectly include this file:



Classes

- class [TileImprovement](#)
A base class for the tile improvement hierarchy.

Enumerations

- enum [TileImprovementType](#) {
SETTLEMENT, DIESEL_GENERATOR, SOLAR_PV, WIND_TURBINE,
TIDAL_TURBINE, WAVE_ENERGY_CONVERTER, ENERGY_STORAGE_SYSTEM, N_TILE_IMPROVEMENT_TYPES
}
An enumeration of the different tile improvement types.

5.16.1 Detailed Description

Header file for the [TileImprovement](#) class.

5.16.2 Enumeration Type Documentation

5.16.2.1 TileImprovementType

enum [TileImprovementType](#)

An enumeration of the different tile improvement types.

Enumerator

SETTLEMENT	A settlement.
DIESEL_GENERATOR	A diesel generator.
SOLAR_PV	A solar PV array.
WIND_TURBINE	A wind turbine.
TIDAL_TURBINE	A tidal turbine.
WAVE_ENERGY_CONVERTER	A wave energy converter.
ENERGY_STORAGE_SYSTEM	An energy storage system.
N_TILE_IMPROVEMENT_TYPES	A simple hack to get the number of elements in TileImprovementType.

```

68         {
69             SETTLEMENT,
70             DIESEL_GENERATOR,
71             SOLAR_PV,
72             WIND_TURBINE,
73             TIDAL_TURBINE,
74             WAVE_ENERGY_CONVERTER,
75             ENERGY_STORAGE_SYSTEM,
76             N_TILE_IMPROVEMENT_TYPES
77 }; /* TileImprovementType */

```

5.17 header/WaveEnergyConverter.h File Reference

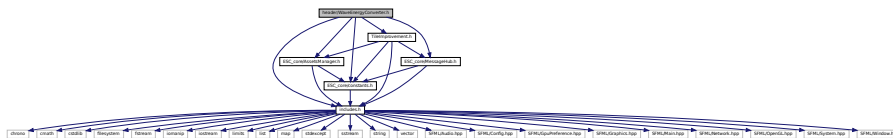
Header file for the [WaveEnergyConverter](#) class.

```

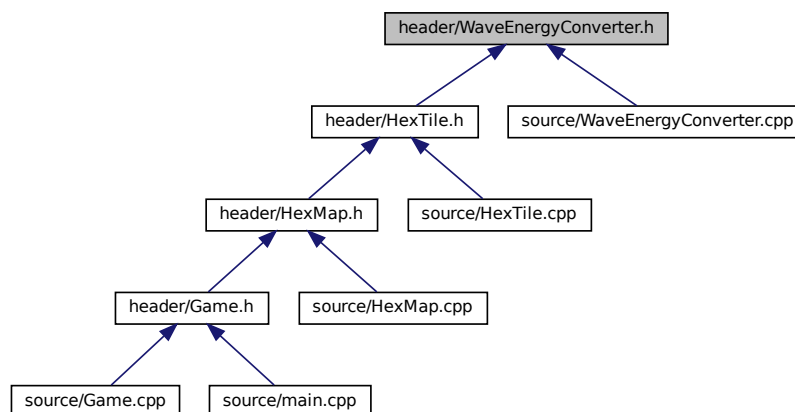
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"

```

Include dependency graph for WaveEnergyConverter.h:



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

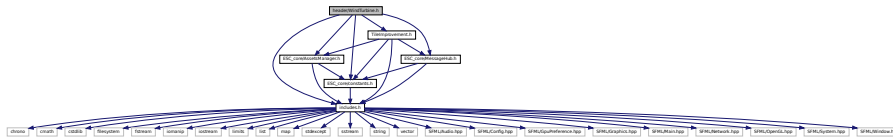
Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

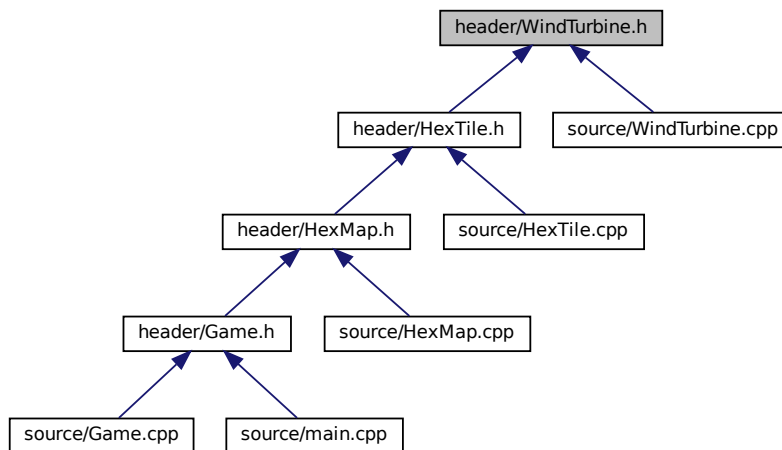
Header file for the `WindTurbine` class.

```
#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:



This graph shows which files directly or indirectly include this file:



Classes

- class **WindTurbine**
*A settlement class (child class of **TileImprovement**).*

5.18.1 Detailed Description

Header file for the `WindTurbine` class.

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__").

```

462 {
463     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
464     error_str += std::to_string(line);
465     error_str += " of ";
466     error_str += file;
467
468     #ifdef _WIN32
469         std::cout << error_str << std::endl;
470     #endif
471
472     throw std::runtime_error(error_str);
473     return;
474 } /* 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.
------------------	---

```

114 {
115     std::cout << "\x1B[33m" << input_str << "\033[0m";
116     return;
117 } /* 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.
------------------	---

```

94 {
95     std::cout << "\x1B[32m" << input_str << "\033[0m";
96     return;
97 } /* 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> .
------------------	---

```
134 {
135     std::cout << "\x1B[31m" << input_str << "\033[0m";
136     return;
137 } /* 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> ").

```
168 {
169     if (fabs(x - y) <= FLOAT_TOLERANCE) {
170         return;
171     }
172
173     std::string error_str = "ERROR: testFloatEquals():\t in ";
174     error_str += file;
175     error_str += "\tline ";
176     error_str += std::to_string(line);
177     error_str += ":\t\n";
178     error_str += std::to_string(x);
179     error_str += " and ";
180     error_str += std::to_string(y);
181     error_str += " are not equal to within +/- ";
182     error_str += std::to_string(FLOAT_TOLERANCE);
183     error_str += "\n";
184
185     #ifdef _WIN32
186         std::cout << error_str << std::endl;
187     #endif
188
189     throw std::runtime_error(error_str);
190     return;
191 } /* 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__").

```
221 {
222     if (x > y) {
223         return;
224     }
225
226     std::string error_str = "ERROR: testGreaterThan():\t in ";
227     error_str += file;
228     error_str += "\tline ";
229     error_str += std::to_string(line);
230     error_str += ":\t\n";
231     error_str += std::to_string(x);
232     error_str += " is not greater than ";
233     error_str += std::to_string(y);
234     error_str += "\n";
235
236     #ifdef _WIN32
237         std::cout << error_str << std::endl;
238     #endif
239
240     throw std::runtime_error(error_str);
241     return;
242 } /* 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__").

```
272 {
273     if (x >= y) {
274         return;
275     }
276
277     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
278     error_str += file;
279     error_str += "\tline ";
280     error_str += std::to_string(line);
281     error_str += ":\t\n";
```

```

282     error_str += std::to_string(x);
283     error_str += " is not greater than or equal to ";
284     error_str += std::to_string(y);
285     error_str += "\n";
286
287     #ifdef _WIN32
288         std::cout << error_str << std::endl;
289     #endif
290
291     throw std::runtime_error(error_str);
292     return;
293 } /* 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__").

```

323 {
324     if (x < y) {
325         return;
326     }
327
328     std::string error_str = "ERROR: testLessThan():\t in ";
329     error_str += file;
330     error_str += "\tline ";
331     error_str += std::to_string(line);
332     error_str += ":\t\n";
333     error_str += std::to_string(x);
334     error_str += " is not less than ";
335     error_str += std::to_string(y);
336     error_str += "\n";
337
338     #ifdef _WIN32
339         std::cout << error_str << std::endl;
340     #endif
341
342     throw std::runtime_error(error_str);
343     return;
344 } /* 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__").

```

374 {
375     if (x <= y) {
376         return;
377     }
378
379     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
380     error_str += file;
381     error_str += "\tline ";
382     error_str += std::to_string(line);
383     error_str += ":\t\n";
384     error_str += std::to_string(x);
385     error_str += " is not less than or equal to ";
386     error_str += std::to_string(y);
387     error_str += "\n";
388
389     #ifdef _WIN32
390         std::cout << error_str << std::endl;
391     #endif
392
393     throw std::runtime_error(error_str);
394     return;
395 } /* 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__").

```

422 {
423     if (statement) {
424         return;
425     }
426
427     std::string error_str = "ERROR: testTruth():\t in ";
428     error_str += file;
429     error_str += "\tline ";
430     error_str += std::to_string(line);
431     error_str += ":\t\n";
432     error_str += "Given statement is not true";
433
434     #ifdef _WIN32
435         std::cout << error_str << std::endl;
436     #endif
437
438     throw std::runtime_error(error_str);
439     return;
440 } /* testTruth() */

```


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.

```
294 {
295     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
296         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
297         "Road To Zero"
298     );
299
300     return render_window_ptr;
301 } /* 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.
---------------------------------	--------------------------------

```
66 {
67     // 1. load font assets
68     assets_manager_ptr->loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
69     assets_manager_ptr->loadFont("assets/fonts/Glass_TTY_VT220.ttf", "Glass_TTY_VT220");
70
71
72     // 2. load tile sheets
73     assets_manager_ptr->loadTexture(
74         "assets/tile_sheets/pine_tree_64x64_1_CC-BY.png",
75         "pine_tree_64x64_1"
76     );
77
78     assets_manager_ptr->loadTexture(
79         "assets/tile_sheets/wheat_64x64_1_CC-BY.png",
80         "wheat_64x64_1"
81     );
82
83     assets_manager_ptr->loadTexture(
84         "assets/tile_sheets/mountain_64x64_1_CC-BY.png",
85         "mountain_64x64_1"
86     );
87
88     assets_manager_ptr->loadTexture(
89         "assets/tile_sheets/water_waves_64x64_1_CC-BY.png",
90         "water_waves_64x64_1"
91     );
92
93     assets_manager_ptr->loadTexture(
94         "assets/tile_sheets/water_shimmer_64x64_1_CC-BY.png",
```

```
95     "water_shimmer_64x64_1"
96 );
97
98 assets_manager_ptr->loadTexture(
99     "assets/tile_sheets/brick_house_64x64_1_CC-BY.png",
100     "brick_house_64x64_1"
101 );
102
103 assets_manager_ptr->loadTexture(
104     "assets/tile_sheets/magnifying_glass_64x64_1_CC-BY.png",
105     "magnifying_glass_64x64_1"
106 );
107
108 assets_manager_ptr->loadTexture(
109     "assets/tile_sheets/exp2_0_CC0.png",
110     "tile clear explosion"
111 );
112
113 assets_manager_ptr->loadTexture(
114     "assets/tile_sheets/emissions_8x8_1_CC-BY.png",
115     "emissions"
116 );
117
118 assets_manager_ptr->loadTexture(
119     "assets/tile_sheets/diesel_generator_64x64_2_CC-BY.png",
120     "diesel generator"
121 );
122
123 assets_manager_ptr->loadTexture(
124     "assets/tile_sheets/solar_PV_64x64_1_CC-BY.png",
125     "solar PV array"
126 );
127
128 assets_manager_ptr->loadTexture(
129     "assets/tile_sheets/wind_turbine_64x64_2_CC-BY.png",
130     "wind turbine"
131 );
132
133 assets_manager_ptr->loadTexture(
134     "assets/tile_sheets/energy_storage_system_64x64_1_CC-BY.png",
135     "energy storage system"
136 );
137
138 assets_manager_ptr->loadTexture(
139     "assets/tile_sheets/tidal_turbine_64x64_2_CC-BY.png",
140     "tidal turbine"
141 );
142
143 assets_manager_ptr->loadTexture(
144     "assets/tile_sheets/wave_energy_converter_64x64_2_CC-BY.png",
145     "wave energy converter"
146 );
147
148
149 // 3. load sounds
150 assets_manager_ptr->loadSound(
151     "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
152     "coin ring"
153 );
154
155 assets_manager_ptr->loadSound(
156     "assets/audio/samples/mixkit-positive-notification-951_MixkitFree.ogg",
157     "positive notification"
158 );
159
160 assets_manager_ptr->loadSound(
161     "assets/audio/samples/mixkit-sci-fi-click-900_MixkitFree.ogg",
162     "sci-fi click"
163 );
164
165 assets_manager_ptr->loadSound(
166     "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932_MixkitFree.ogg",
167     "insufficient credits"
168 );
169
170 assets_manager_ptr->loadSound(
171     "assets/audio/samples/mixkit-data-scanner-2487_MixkitFree.ogg",
172     "resource assessment"
173 );
174
175 assets_manager_ptr->loadSound(
176     "assets/audio/samples/mixkit-interface-click-1126_MixkitFree.ogg",
177     "console string print"
178 );
179
180 assets_manager_ptr->loadSound(
181     "assets/audio/samples/mixkit-video-game-retro-click-237_MixkitFree.ogg",
```

```

182     "resource overlay toggle on"
183 );
184
185 assets_manager_ptr->loadSound(
186     "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED_MixkitFree.ogg",
187     "resource overlay toggle off"
188 );
189
190 assets_manager_ptr->loadSound(
191     "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703_MixkitFree.ogg",
192     "clear mountains tile"
193 );
194
195 assets_manager_ptr->loadSound(
196     "assets/audio/samples/mixkit-arcade-game-explosion-2759_MixkitFree.ogg",
197     "clear non-mountains tile"
198 );
199
200 assets_manager_ptr->loadSound(
201     "assets/audio/samples/mixkit-electronic-retro-block-hit-2185_MixkitFree.ogg",
202     "place improvement"
203 );
204
205 assets_manager_ptr->loadSound(
206     "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED_MixkitFree.ogg",
207     "build menu open"
208 );
209
210 assets_manager_ptr->loadSound(
211     "assets/audio/samples/mixkit-video-game-lock-2851_MixkitFree.ogg",
212     "build menu close"
213 );
214
215 assets_manager_ptr->loadSound(
216     "assets/audio/samples/mixkit-jump-into-the-water-1180_MixkitFree.ogg",
217     "splash"
218 );
219
220 assets_manager_ptr->loadSound(
221     "assets/audio/samples/505316__nuncaconoci__diesel_CC0.ogg",
222     "diesel running"
223 );
224
225 assets_manager_ptr->loadSound(
226     "assets/audio/samples/33460__pempi__320d_2_CC-BY.ogg",
227     "diesel start"
228 );
229
230 assets_manager_ptr->loadSound(
231     "assets/audio/samples/132724__andy_gardner__wind-turbine-blades_CC-BY.ogg",
232     "wind turbine running"
233 );
234
235 assets_manager_ptr->loadSound(
236     "assets/audio/samples/58416__darren1979__oceanwaves_CC-SAMPLING.ogg",
237     "ocean waves"
238 );
239
240 assets_manager_ptr->loadSound(
241     "assets/audio/samples/369927__mephisto_egmont__water-flowing-in-tubes_CC-BY.ogg",
242     "water flow"
243 );
244
245 assets_manager_ptr->loadSound(
246     "assets/audio/samples/647663__jotraing__electric-train-motor-idle-loop-new-generation-rollingstock_CC0.ogg",
247     "energy storage system idle"
248 );
249
250 assets_manager_ptr->loadSound(
251     "assets/audio/samples/mixkit-epic-futuristic-movie-accent-2913_MixkitFree.ogg",
252     "game title screen"
253 );
254
255 assets_manager_ptr->loadSound(
256     "assets/audio/samples/mixkit-calm-park-with-people-and-children_MixkitFree.ogg",
257     "people and children"
258 );
259
260
261 // 4. load tracks
262 assets_manager_ptr->loadTrack(
263     "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
264     "Tree Star Moon - Dobranoc"
265 );
266
267 assets_manager_ptr->loadTrack(

```



```

268         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
269         "Tree Star Moon - Lighthouse"
270     );
271
272     assets_manager_ptr->loadTrack (
273         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
274         "Tree Star Moon - Sky Farm"
275     );
276
277     return;
278 } /* loadAssets() */

```

5.28.2.3 main()

```

1  int main (
2      int argc,
3      char ** argv )
4  {
5      // 1. load assets
6      AssetsManager assets_manager;
7      loadAssets(&assets_manager);
8
9      // 2. construct render window
10     sf::RenderWindow* render_window_ptr = constructRenderWindow();
11
12     // 3. start game loop
13     bool quit_game = false;
14     assets_manager.playTrack();
15
16     while (not quit_game) {
17         Game game(render_window_ptr, &assets_manager);
18         quit_game = game.run();
19     }
20
21     // 4. clean up
22     render_window_ptr->close();
23     delete render_window_ptr;
24
25     return 0;
26 } /* main() */

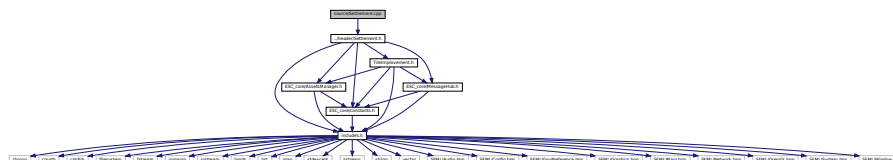
```

5.29 source/Settlement.cpp File Reference

Implementation file for the **Settlement** class.

```
#include "../header/Settlement.h"
```

Include dependency graph for Settlement.cpp:



5.29.1 Detailed Description

Implementation file for the **Settlement** class.

A base class for the tile improvement hierarchy.

Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL <https://www.sfml-dev.org/>. 199

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 198

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 39, 47, 92, 142, 148, 154, 161, 172, 178

Index

- __assembleHexMap
 - HexMap, [69](#)
- __assessNeighbours
 - HexMap, [69](#)
- __buildDieselGenerator
 - HexTile, [94](#)
- __buildDrawOrderVector
 - HexMap, [70](#)
- __buildEnergyStorage
 - HexTile, [94](#)
- __buildSettlement
 - HexTile, [95](#)
- __buildSolarPV
 - HexTile, [95](#)
- __buildTidalTurbine
 - HexTile, [96](#)
- __buildWaveEnergyConverter
 - HexTile, [96](#)
- __buildWindTurbine
 - HexTile, [97](#)
- __clearDecoration
 - HexTile, [98](#)
- __closeBuildMenu
 - HexTile, [98](#)
- __closeProductionMenu
 - TileImprovement, [162](#)
- __draw
 - Game, [54](#)
- __drawConsoleScreenFrame
 - ContextMenu, [22](#)
- __drawConsoleText
 - ContextMenu, [23](#)
- __drawFrameClockOverlay
 - Game, [54](#)
- __drawHUD
 - Game, [54](#)
- __drawVisualScreenFrame
 - ContextMenu, [24](#)
- __enforceOceanContinuity
 - HexMap, [70](#)
- __getMajorityTileType
 - HexMap, [71](#)
- __getNeighboursVector
 - HexMap, [72](#)
- __getNoise
 - HexMap, [73](#)
- __getSelectedTile
 - HexMap, [74](#)
- __getTileCoordsSubstring
 - HexTile, [98](#)
- __getTileImprovementSubstring
 - HexTile, [99](#)
- __getTileOptionsSubstring
 - HexTile, [99](#)
- __getTileResourceSubstring
 - HexTile, [101](#)
- __getTileTypeSubstring
 - HexTile, [101](#)
- __getValidMapIndexPositions
 - HexMap, [75](#)
- __handleKeyPressEvents
 - ContextMenu, [24](#)
 - DieselGenerator, [40](#)
 - EnergyStorageSystem, [48](#)
 - Game, [56](#)
 - HexMap, [76](#)
 - HexTile, [102](#)
 - Settlement, [143](#)
 - SolarPV, [149](#)
 - TidalTurbine, [155](#)
 - TileImprovement, [163](#)
 - WaveEnergyConverter, [173](#)
 - WindTurbine, [179](#)
- __handleMouseButtonEvents
 - ContextMenu, [25](#)
 - DieselGenerator, [40](#)
 - EnergyStorageSystem, [48](#)
 - Game, [56](#)
 - HexMap, [76](#)
 - HexTile, [106](#)
 - Settlement, [143](#)
 - SolarPV, [150](#)
 - TidalTurbine, [155](#)
 - TileImprovement, [163](#)
 - WaveEnergyConverter, [173](#)
 - WindTurbine, [179](#)
- __insufficientCreditsAlarm
 - Game, [57](#)
- __isClicked
 - HexTile, [107](#)
- __isLakeTouchingOcean
 - HexMap, [77](#)
- __layTiles
 - HexMap, [77](#)
- __loadSoundBuffer
 - AssetsManager, [9](#)
- __openBuildMenu
 - HexTile, [107](#)

- __openProductionMenu
 - TileImprovement, 163
- __procedurallyGenerateTileResources
 - HexMap, 79
- __procedurallyGenerateTileTypes
 - HexMap, 80
- __processEvent
 - Game, 58
- __processMessage
 - Game, 58
- __scrapImprovement
 - HexTile, 107
- __sendAssessNeighboursMessage
 - HexTile, 108
- __sendCreditsSpentMessage
 - HexTile, 108
- __sendGameStateMessage
 - Game, 60
- __sendGameStateRequest
 - HexTile, 109
- __sendInsufficientCreditsMessage
 - HexTile, 109
- __sendNoTileSelectedMessage
 - HexMap, 80
- __sendQuitGameMessage
 - ContextMenu, 25
- __sendRestartGameMessage
 - ContextMenu, 25
- __sendTileSelectedMessage
 - HexTile, 109
- __sendTileStateMessage
 - HexTile, 109
- __sendUpdateGamePhaseMessage
 - HexTile, 110
- __setConsoleState
 - ContextMenu, 26
- __setConsoleString
 - ContextMenu, 26
- __setIsSelected
 - HexTile, 110
- __setResourceText
 - HexTile, 111
- __setUpBuildMenu
 - HexTile, 112
- __setUpBuildOption
 - HexTile, 113
- __setUpConsoleScreen
 - ContextMenu, 27
- __setUpConsoleScreenFrame
 - ContextMenu, 27
- __setUpDieselGeneratorBuildOption
 - HexTile, 114
- __setUpEnergyStorageSystemBuildOption
 - HexTile, 114
- __setUpGlassScreen
 - HexMap, 81
- __setUpMagnifyingGlassSprite
 - HexTile, 115
- __setUpMenuFrame
 - ContextMenu, 29
- __setUpNodeSprite
 - HexTile, 115
- __setUpProductionMenu
 - TileImprovement, 164
- __setUpResourceChipSprite
 - HexTile, 115
- __setUpSelectOutlineSprite
 - HexTile, 116
- __setUpSolarPVBuildOption
 - HexTile, 116
- __setUpTidalTurbineBuildOption
 - HexTile, 117
- __setUpTileExplosionReel
 - HexTile, 117
- __setUpTileImprovementSpriteAnimated
 - DieselGenerator, 41
 - TidalTurbine, 156
 - WaveEnergyConverter, 174
 - WindTurbine, 179
- __setUpTileImprovementSpriteStatic
 - EnergyStorageSystem, 48
 - Settlement, 143
 - SolarPV, 150
- __setUpTileSprite
 - HexTile, 118
- __setUpVisualScreen
 - ContextMenu, 30
- __setUpVisualScreenFrame
 - ContextMenu, 30
- __setUpWaveEnergyConverterBuildOption
 - HexTile, 118
- __setUpWindTurbineBuildOption
 - HexTile, 118
- __smoothTileTypes
 - HexMap, 81
- __toggleFrameClockOverlay
 - Game, 61
- ~AssetsManager
 - AssetsManager, 8
- ~ContextMenu
 - ContextMenu, 22
- ~DieselGenerator
 - DieselGenerator, 40
- ~EnergyStorageSystem
 - EnergyStorageSystem, 47
- ~Game
 - Game, 53
- ~HexMap
 - HexMap, 69
- ~HexTile
 - HexTile, 93
- ~MessageHub
 - MessageHub, 134
- ~Settlement
 - Settlement, 142
- ~SolarPV

- SolarPV, [149](#)
- ~TidalTurbine
 - TidalTurbine, [155](#)
- ~TileImprovement
 - TileImprovement, [162](#)
- ~WaveEnergyConverter
 - WaveEnergyConverter, [173](#)
- ~WindTurbine
 - WindTurbine, [178](#)
- ABOVE_AVERAGE
 - HexTile.h, [210](#)
- addChannel
 - MessageHub, [135](#)
- assess
 - HexMap, [81](#)
 - HexTile, [119](#)
- assets_manager_ptr
 - ContextMenu, [33](#)
 - Game, [62](#)
 - HexMap, [85](#)
 - HexTile, [126](#)
 - TileImprovement, [167](#)
- 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, [210](#)
- BELOW_AVERAGE
 - HexTile.h, [210](#)
- bool_payload
 - Message, [132](#)
- border_tiles_vec
 - HexMap, [85](#)
- build_menu_backing
 - HexTile, [126](#)
- build_menu_backing_text
 - HexTile, [126](#)
- build_menu_open
 - HexTile, [126](#)
- build_menu_options_text_vec
 - HexTile, [126](#)
- build_menu_options_vec
 - HexTile, [126](#)
- BUILD_SETTLEMENT
 - Game.h, [208](#)
- BUILD_SETTLEMENT_COST
 - constants.h, [192](#)
- capacity_kW
 - DieselGenerator, [44](#)
- channel
 - Message, [132](#)
- clear
 - AssetsManager, [10](#)
 - HexMap, [82](#)
 - MessageHub, [135](#)
- CLEAR_FOREST_COST
 - constants.h, [192](#)
- CLEAR_MOUNTAINS_COST
 - constants.h, [192](#)
- CLEAR_PLAINS_COST
 - constants.h, [193](#)
- clearMessages
 - MessageHub, [136](#)
- clock
 - Game, [62](#)
- CO2E_KG_PER_LITRE_DIESEL
 - constants.h, [193](#)
- 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, [184](#)
- constants.h
 - BUILD_SETTLEMENT_COST, [192](#)
 - CLEAR_FOREST_COST, [192](#)
 - CLEAR_MOUNTAINS_COST, [192](#)
 - CLEAR_PLAINS_COST, [193](#)

- CO2E_KG_PER_LITRE_DIESEL, 193
- DIESEL_GENERATOR_BUILD_COST, 193
- EMISSIONS_LIFETIME_LIMIT_TONNES, 193
- ENERGY_STORAGE_SYSTEM_BUILD_COST, 193
- FLOAT_TOLERANCE, 193
- FOREST_GREEN, 190
- FRAMES_PER_SECOND, 194
- GAME_CHANNEL, 194
- GAME_HEIGHT, 194
- GAME_STATE_CHANNEL, 194
- GAME_WIDTH, 194
- HEX_MAP_CHANNEL, 194
- LAKE_BLUE, 190
- MENU_FRAME_GREY, 190
- MONOCHROME_SCREEN_BACKGROUND, 190
- MONOCHROME_TEXT_AMBER, 190
- MONOCHROME_TEXT_GREEN, 191
- MONOCHROME_TEXT_RED, 191
- MOUNTAINS_GREY, 191
- NO_TILE_SELECTED_CHANNEL, 195
- OCEAN_BLUE, 191
- PLAINS_YELLOW, 191
- RESOURCE_ASSESSMENT_COST, 195
- RESOURCE_CHIP_GREY, 192
- SCRAP_COST, 195
- SECONDS_PER_FRAME, 195
- SECONDS_PER_MONTH, 195
- SECONDS_PER_YEAR, 195
- SOLAR_PV_BUILD_COST, 196
- SOLAR_PV_WATER_BUILD_MULTIPLIER, 196
- STARTING_CREDITS, 196
- STARTING_POPULATION, 196
- TIDAL_TURBINE_BUILD_COST, 196
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES, ContextMenu.h 196
- TILE_SELECTED_CHANNEL, 197
- TILE_STATE_CHANNEL, 197
- TILE_TYPE_CUMULATIVE_PROBABILITIES, 197
- VISUAL_SCREEN_FRAME_GREY, 192
- WAVE_ENERGY_CONVERTER_BUILD_COST, 197
- WIND_TURBINE_BUILD_COST, 197
- WIND_TURBINE_WATER_BUILD_MULTIPLIER, 198
- constructRenderWindow
 - main.cpp, 228
- context_menu_ptr
 - Game, 62
- 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
- credits
 - Game, 62
 - HexTile, 127
 - TileImprovement, 167
- cumulative_emissions_tonnes
 - Game, 62
- current_track
 - AssetsManager, 18
- decorateTile
 - HexTile, 119
- decoration_cleared
 - HexTile, 127
- demand_MWh
 - Game, 63
- DIESEL_GENERATOR
 - TileImprovement.h, 215
- DIESEL_GENERATOR_BUILD_COST

- constants.h, 193
- DieselGenerator, 37
 - __handleKeyPressEvents, 40
 - __handleMouseButtonEvents, 40
 - __setUpTileImprovementSpriteAnimated, 41
 - ~DieselGenerator, 40
 - capacity_kW, 44
 - DieselGenerator, 39
 - draw, 41
 - getTileOptionsSubstring, 42
 - max_production_MWh, 44
 - processEvent, 43
 - processMessage, 43
 - production_MWh, 44
 - smoke_da, 44
 - smoke_dx, 44
 - smoke_dy, 44
 - smoke_prob, 45
 - smoke_sprite_list, 45
- double_payload
 - Message, 133
- draw
 - ContextMenu, 31
 - DieselGenerator, 41
 - EnergyStorageSystem, 49
 - HexMap, 82
 - HexTile, 121
 - Settlement, 144
 - SolarPV, 151
 - TidalTurbine, 156
 - TileImprovement, 164
 - WaveEnergyConverter, 174
 - WindTurbine, 180
- draw_explosion
 - HexTile, 127
- EMISSIONS_LIFETIME_LIMIT_TONNES
 - constants.h, 193
- ENERGY_STORAGE_SYSTEM
 - TileImprovement.h, 215
- ENERGY_STORAGE_SYSTEM_BUILD_COST
 - constants.h, 193
- EnergyStorageSystem, 45
 - __handleKeyPressEvents, 48
 - __handleMouseButtonEvents, 48
 - __setUpTileImprovementSpriteStatic, 48
 - ~EnergyStorageSystem, 47
 - draw, 49
 - EnergyStorageSystem, 47
 - getTileOptionsSubstring, 49
 - processEvent, 50
 - processMessage, 50
- event
 - Game, 63
- event_ptr
 - ContextMenu, 35
 - HexMap, 85
 - HexTile, 127
 - TileImprovement, 167
- expectedErrorNotDetected
 - testing_utils.cpp, 220
 - testing_utils.h, 201
- explosion_frame
 - HexTile, 127
- explosion_sprite_reel
 - HexTile, 127
- FLOAT_TOLERANCE
 - constants.h, 193
- font_map
 - AssetsManager, 18
- FOREST
 - HexTile.h, 211
- FOREST_GREEN
 - constants.h, 190
- frame
 - ContextMenu, 35
 - Game, 63
 - HexMap, 85
 - HexTile, 128
 - TileImprovement, 168
- FRAMES_PER_SECOND
 - constants.h, 194
- Game, 51
 - __draw, 54
 - __drawFrameClockOverlay, 54
 - __drawHUD, 54
 - __handleKeyPressEvents, 56
 - __handleMouseButtonEvents, 56
 - __insufficientCreditsAlarm, 57
 - __processEvent, 58
 - __processMessage, 58
 - __sendGameStateMessage, 60
 - __toggleFrameClockOverlay, 61
 - ~Game, 53
 - assets_manager_ptr, 62
 - clock, 62
 - context_menu_ptr, 62
 - credits, 62
 - cumulative_emissions_tonnes, 62
 - demand_MWh, 63
 - event, 63
 - frame, 63
 - Game, 53
 - game_loop_broken, 63
 - game_phase, 63
 - hex_map_ptr, 63
 - message_hub, 64
 - month, 64
 - population, 64
 - quit_game, 64
 - render_window_ptr, 64
 - run, 61
 - show_frame_clock_overlay, 64
 - time_since_start_s, 65
 - turn, 65
 - year, 65

Game.h
 BUILD_SETTLEMENT, 208
 GamePhase, 207
 LOSS_CREDITS, 208
 LOSS_DEMAND, 208
 LOSS_EMISSIONS, 208
 N_GAME_PHASES, 208
 SYSTEM_MANAGEMENT, 208
 VICTORY, 208
 GAME_CHANNEL
 constants.h, 194
 GAME_HEIGHT
 constants.h, 194
 game_loop_broken
 Game, 63
 game_menu_up
 ContextMenu, 35
 game_phase
 Game, 63
 HexTile, 128
 TileImprovement, 168
 GAME_STATE_CHANNEL
 constants.h, 194
 GAME_WIDTH
 constants.h, 194
 GamePhase
 Game.h, 207
 getCurrentTrackKey
 AssetsManager, 11
 getFont
 AssetsManager, 11
 getSound
 AssetsManager, 12
 getSoundBuffer
 AssetsManager, 12
 getTexture
 AssetsManager, 13
 getTileOptionsSubstring
 DieselGenerator, 42
 EnergyStorageSystem, 49
 Settlement, 145
 SolarPV, 151
 TidalTurbine, 157
 TileImprovement, 166
 WaveEnergyConverter, 175
 WindTurbine, 180
 getTrackStatus
 AssetsManager, 13
 glass_screen
 HexMap, 85
 GOOD
 HexTile.h, 210
 has_improvement
 HexTile, 128
 hasTraffic
 MessageHub, 136
 header/ContextMenu.h, 183
 header/DieselGenerator.h, 184
 header/EnergyStorageSystem.h, 185
 header/ESC_core/AssetsManager.h, 186
 header/ESC_core/constants.h, 187
 header/ESC_core/doxygen_cite.h, 198
 header/ESC_core/includes.h, 198
 header/ESC_core/MessageHub.h, 199
 header/ESC_core/testing_utils.h, 200
 header/Game.h, 206
 header/HexMap.h, 208
 header/HexTile.h, 209
 header/Settlement.h, 211
 header/SolarPV.h, 212
 header/TidalTurbine.h, 213
 header/TileImprovement.h, 214
 header/WaveEnergyConverter.h, 216
 header/WindTurbine.h, 217
 health
 TileImprovement, 168
 hex_draw_order_vec
 HexMap, 86
 hex_map
 HexMap, 86
 HEX_MAP_CHANNEL
 constants.h, 194
 hex_map_ptr
 Game, 63
 HexMap, 65
 __assembleHexMap, 69
 __assessNeighbours, 69
 __buildDrawOrderVector, 70
 __enforceOceanContinuity, 70
 __getMajorityTileType, 71
 __getNeighboursVector, 72
 __getNoise, 73
 __getSelectedTile, 74
 __getValidMapIndexPositions, 75
 __handleKeyPressEvents, 76
 __handleMouseButtonEvents, 76
 __isLakeTouchingOcean, 77
 __layTiles, 77
 __procedurallyGenerateTileResources, 79
 __procedurallyGenerateTileTypes, 80
 __sendNoTileSelectedMessage, 80
 __setUpGlassScreen, 81
 __smoothTileTypes, 81
 ~HexMap, 69
 assess, 81
 assets_manager_ptr, 85
 border_tiles_vec, 85
 clear, 82
 draw, 82
 event_ptr, 85
 frame, 85
 glass_screen, 85
 hex_draw_order_vec, 86
 hex_map, 86
 HexMap, 68
 message_hub_ptr, 86

- n_layers, 86
- n_tiles, 86
- position_x, 86
- position_y, 87
- processEvent, 83
- processMessage, 83
- render_window_ptr, 87
- reroll, 84
- show_resource, 87
- tile_position_x_vec, 87
- tile_position_y_vec, 87
- tile_selected, 87
- toggleResourceOverlay, 84
- HexTile, 88
 - __buildDieselGenerator, 94
 - __buildEnergyStorage, 94
 - __buildSettlement, 95
 - __buildSolarPV, 95
 - __buildTidalTurbine, 96
 - __buildWaveEnergyConverter, 96
 - __buildWindTurbine, 97
 - __clearDecoration, 98
 - __closeBuildMenu, 98
 - __getTileCoordsSubstring, 98
 - __getTileImprovementSubstring, 99
 - __getTileOptionsSubstring, 99
 - __getTileResourceSubstring, 101
 - __getTileTypeSubstring, 101
 - __handleKeyPressEvents, 102
 - __handleMouseButtonEvents, 106
 - __isClicked, 107
 - __openBuildMenu, 107
 - __scrapImprovement, 107
 - __sendAssessNeighboursMessage, 108
 - __sendCreditsSpentMessage, 108
 - __sendGameStateRequest, 109
 - __sendInsufficientCreditsMessage, 109
 - __sendTileSelectedMessage, 109
 - __sendTileStateMessage, 109
 - __sendUpdateGamePhaseMessage, 110
 - __setIsSelected, 110
 - __setResourceText, 111
 - __setUpBuildMenu, 112
 - __setUpBuildOption, 113
 - __setUpDieselGeneratorBuildOption, 114
 - __setUpEnergyStorageSystemBuildOption, 114
 - __setUpMagnifyingGlassSprite, 115
 - __setUpNodeSprite, 115
 - __setUpResourceChipSprite, 115
 - __setUpSelectOutlineSprite, 116
 - __setUpSolarPVBuildOption, 116
 - __setUpTidalTurbineBuildOption, 117
 - __setUpTileExplosionReel, 117
 - __setUpTileSprite, 118
 - __setUpWaveEnergyConverterBuildOption, 118
 - __setUpWindTurbineBuildOption, 118
 - ~HexTile, 93
 - assess, 119
 - assets_manager_ptr, 126
 - build_menu_backing, 126
 - build_menu_backing_text, 126
 - build_menu_open, 126
 - build_menu_options_text_vec, 126
 - build_menu_options_vec, 126
 - credits, 127
 - decorateTile, 119
 - decoration_cleared, 127
 - draw, 121
 - draw_explosion, 127
 - event_ptr, 127
 - explosion_frame, 127
 - explosion_sprite_reel, 127
 - frame, 128
 - game_phase, 128
 - has_improvement, 128
 - HexTile, 92
 - is_selected, 128
 - magnifying_glass_sprite, 128
 - major_radius, 128
 - message_hub_ptr, 129
 - minor_radius, 129
 - node_sprite, 129
 - position_x, 129
 - position_y, 129
 - processEvent, 122
 - processMessage, 122
 - render_window_ptr, 129
 - resource_assessed, 130
 - resource_assessment, 130
 - resource_chip_sprite, 130
 - resource_text, 130
 - select_outline_sprite, 130
 - setTileResource, 123
 - setTileType, 124
 - show_node, 130
 - show_resource, 131
 - tile_decoration_sprite, 131
 - tile_improvement_ptr, 131
 - tile_resource, 131
 - tile_sprite, 131
 - tile_type, 131
 - toggleResourceOverlay, 125
- HexTile.h
 - ABOVE_AVERAGE, 210
 - AVERAGE, 210
 - BELOW_AVERAGE, 210
 - FOREST, 211
 - GOOD, 210
 - LAKE, 211
 - MOUNTAINS, 211
 - N_TILE_RESOURCES, 210
 - N_TILE_TYPES, 211
 - NONE_TYPE, 211
 - OCEAN, 211
 - PLAINS, 211
 - POOR, 210

- TileResource, 210
- TileType, 210
- int_payload
 - Message, 133
- is_running
 - TileImprovement, 168
- is_selected
 - HexTile, 128
 - TileImprovement, 168
- isEmpty
 - MessageHub, 136
- just_built
 - TileImprovement, 168
- LAKE
 - HexTile.h, 211
- LAKE_BLUE
 - constants.h, 190
- loadAssets
 - main.cpp, 228
- loadFont
 - AssetsManager, 14
- loadSound
 - AssetsManager, 14
- loadTexture
 - AssetsManager, 15
- loadTrack
 - AssetsManager, 16
- LOSS_CREDITS
 - Game.h, 208
- LOSS_DEMAND
 - Game.h, 208
- LOSS_EMISSIONS
 - Game.h, 208
- magnifying_glass_sprite
 - HexTile, 128
- main
 - main.cpp, 231
- main.cpp
 - constructRenderWindow, 228
 - loadAssets, 228
 - main, 231
- major_radius
 - HexTile, 128
- max_production_MWh
 - DieselGenerator, 44
- MENU
 - ContextMenu.h, 184
- menu_frame
 - ContextMenu, 35
- MENU_FRAME_GREY
 - constants.h, 190
- Message, 132
 - bool_payload, 132
 - channel, 132
 - double_payload, 133
 - int_payload, 133
 - string_payload, 133
 - subject, 133
- message_hub
 - Game, 64
- message_hub_ptr
 - ContextMenu, 35
 - HexMap, 86
 - HexTile, 129
 - TileImprovement, 169
- message_map
 - MessageHub, 139
- MessageHub, 133
 - ~MessageHub, 134
 - addChannel, 135
 - clear, 135
 - clearMessages, 136
 - hasTraffic, 136
 - isEmpty, 136
 - message_map, 139
 - MessageHub, 134
 - popMessage, 137
 - receiveMessage, 137
 - removeChannel, 138
 - sendMessage, 139
- minor_radius
 - HexTile, 129
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, 190
- MONOCHROME_TEXT_AMBER
 - constants.h, 190
- MONOCHROME_TEXT_GREEN
 - constants.h, 191
- MONOCHROME_TEXT_RED
 - constants.h, 191
- month
 - Game, 64
- MOUNTAINS
 - HexTile.h, 211
- MOUNTAINS_GREY
 - constants.h, 191
- N_CONSOLE_STATES
 - ContextMenu.h, 184
- N_GAME_PHASES
 - Game.h, 208
- n_layers
 - HexMap, 86
- N_TILE_IMPROVEMENT_TYPES
 - TileImprovement.h, 215
- N_TILE_RESOURCES
 - HexTile.h, 210
- N_TILE_TYPES
 - HexTile.h, 211
- n_tiles
 - HexMap, 86
- nextTrack
 - AssetsManager, 16
- NO_TILE_SELECTED_CHANNEL

- constants.h, 195
- node_sprite
 - HexTile, 129
- NONE_STATE
 - ContextMenu.h, 184
- NONE_TYPE
 - HexTile.h, 211
- OCEAN
 - HexTile.h, 211
- OCEAN_BLUE
 - constants.h, 191
- pauseTrack
 - AssetsManager, 17
- PLAINS
 - HexTile.h, 211
- PLAINS_YELLOW
 - constants.h, 191
- playTrack
 - AssetsManager, 17
- POOR
 - HexTile.h, 210
- popMessage
 - MessageHub, 137
- population
 - Game, 64
- position_x
 - ContextMenu, 36
 - HexMap, 86
 - HexTile, 129
 - TileImprovement, 169
- position_y
 - ContextMenu, 36
 - HexMap, 87
 - HexTile, 129
 - TileImprovement, 169
- previousTrack
 - AssetsManager, 17
- printGold
 - testing_utils.cpp, 221
 - testing_utils.h, 202
- printGreen
 - testing_utils.cpp, 221
 - testing_utils.h, 202
- printRed
 - testing_utils.cpp, 221
 - testing_utils.h, 202
- processEvent
 - ContextMenu, 32
 - DieselGenerator, 43
 - EnergyStorageSystem, 50
 - HexMap, 83
 - HexTile, 122
 - Settlement, 145
 - SolarPV, 152
 - TidalTurbine, 157
 - TileImprovement, 166
 - WaveEnergyConverter, 175
 - WindTurbine, 181
- processMessage
 - ContextMenu, 32
 - DieselGenerator, 43
 - EnergyStorageSystem, 50
 - HexMap, 83
 - HexTile, 122
 - Settlement, 145
 - SolarPV, 152
 - TidalTurbine, 158
 - TileImprovement, 166
 - WaveEnergyConverter, 176
 - WindTurbine, 181
- production_menu_backing
 - TileImprovement, 169
- production_menu_backing_text
 - TileImprovement, 169
- production_menu_open
 - TileImprovement, 169
- production_MWh
 - DieselGenerator, 44
- quit_game
 - Game, 64
- READY
 - ContextMenu.h, 184
- receiveMessage
 - MessageHub, 137
- removeChannel
 - MessageHub, 138
- render_window_ptr
 - ContextMenu, 36
 - Game, 64
 - HexMap, 87
 - HexTile, 129
 - TileImprovement, 170
- reroll
 - HexMap, 84
- resource_assessed
 - HexTile, 130
- resource_assessment
 - HexTile, 130
- RESOURCE_ASSESSMENT_COST
 - constants.h, 195
- RESOURCE_CHIP_GREY
 - constants.h, 192
- resource_chip_sprite
 - HexTile, 130
- resource_text
 - HexTile, 130
- run
 - Game, 61
- SCRAP_COST
 - constants.h, 195
- SECONDS_PER_FRAME
 - constants.h, 195
- SECONDS_PER_MONTH

- constants.h, 195
- SECONDS_PER_YEAR
 - constants.h, 195
- select_outline_sprite
 - HexTile, 130
- sendMessage
 - MessageHub, 139
- setIsSelected
 - TileImprovement, 166
- setTileResource
 - HexTile, 123
- setTileType
 - HexTile, 124
- SETTLEMENT
 - TileImprovement.h, 215
- Settlement, 140
 - __handleKeyPressEvents, 143
 - __handleMouseButtonEvents, 143
 - __setUpTileImprovementSpriteStatic, 143
 - ~Settlement, 142
 - draw, 144
 - getTileOptionsSubstring, 145
 - processEvent, 145
 - processMessage, 145
 - Settlement, 141
 - smoke_da, 146
 - smoke_dx, 146
 - smoke_dy, 146
 - smoke_prob, 146
 - smoke_sprite_list, 146
- show_frame_clock_overlay
 - Game, 64
- show_node
 - HexTile, 130
- show_resource
 - HexMap, 87
 - HexTile, 131
- smoke_da
 - DieselGenerator, 44
 - Settlement, 146
- smoke_dx
 - DieselGenerator, 44
 - Settlement, 146
- smoke_dy
 - DieselGenerator, 44
 - Settlement, 146
- smoke_prob
 - DieselGenerator, 45
 - Settlement, 146
- smoke_sprite_list
 - DieselGenerator, 45
 - Settlement, 146
- SOLAR_PV
 - TileImprovement.h, 215
- SOLAR_PV_BUILD_COST
 - constants.h, 196
- SOLAR_PV_WATER_BUILD_MULTIPLIER
 - constants.h, 196
- SolarPV, 147
 - __handleKeyPressEvents, 149
 - __handleMouseButtonEvents, 150
 - __setUpTileImprovementSpriteStatic, 150
 - ~SolarPV, 149
 - draw, 151
 - getTileOptionsSubstring, 151
 - processEvent, 152
 - processMessage, 152
 - SolarPV, 148
- sound_map
 - AssetsManager, 18
- soundbuffer_map
 - AssetsManager, 18
- source/ContextMenu.cpp, 218
- source/DieselGenerator.cpp, 218
- source/EnergyStorageSystem.cpp, 218
- source/ESC_core/AssetsManager.cpp, 219
- source/ESC_core/MessageHub.cpp, 219
- source/ESC_core/testing_utils.cpp, 220
- source/Game.cpp, 226
- source/HexMap.cpp, 226
- source/HexTile.cpp, 227
- source/main.cpp, 227
- source/Settlement.cpp, 231
- source/SolarPV.cpp, 232
- source/TidalTurbine.cpp, 232
- source/TileImprovement.cpp, 232
- source/WaveEnergyConverter.cpp, 233
- source/WindTurbine.cpp, 233
- STARTING_CREDITS
 - constants.h, 196
- STARTING_POPULATION
 - constants.h, 196
- stopTrack
 - AssetsManager, 17
- string_payload
 - Message, 133
- subject
 - Message, 133
- SYSTEM_MANAGEMENT
 - Game.h, 208
- testFloatEquals
 - testing_utils.cpp, 222
 - testing_utils.h, 203
- testGreaterThan
 - testing_utils.cpp, 222
 - testing_utils.h, 203
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 223
 - testing_utils.h, 204
- testing_utils.cpp
 - expectedErrorNotDetected, 220
 - printGold, 221
 - printGreen, 221
 - printRed, 221
 - testFloatEquals, 222
 - testGreaterThan, 222

- testGreaterThanOrEqualTo, 223
- testLessThan, 224
- testLessThanOrEqualTo, 224
- testTruth, 225
- testing_utils.h
 - expectedErrorNotDetected, 201
 - printGold, 202
 - printGreen, 202
 - printRed, 202
 - testFloatEquals, 203
 - testGreaterThan, 203
 - testGreaterThanOrEqualTo, 204
 - testLessThan, 205
 - testLessThanOrEqualTo, 205
 - testTruth, 206
- testLessThan
 - testing_utils.cpp, 224
 - testing_utils.h, 205
- testLessThanOrEqualTo
 - testing_utils.cpp, 224
 - testing_utils.h, 205
- testTruth
 - testing_utils.cpp, 225
 - testing_utils.h, 206
- texture_map
 - AssetsManager, 18
- TIDAL_TURBINE
 - TileImprovement.h, 215
- TIDAL_TURBINE_BUILD_COST
 - constants.h, 196
- TidalTurbine, 153
 - __handleKeyPressEvents, 155
 - __handleMouseButtonEvents, 155
 - __setUpTileImprovementSpriteAnimated, 156
 - ~TidalTurbine, 155
 - draw, 156
 - getTileOptionsSubstring, 157
 - processEvent, 157
 - processMessage, 158
 - TidalTurbine, 154
- TILE
 - ContextMenu.h, 184
- tile_decoration_sprite
 - HexTile, 131
- tile_improvement_ptr
 - HexTile, 131
- tile_improvement_sprite_animated
 - TileImprovement, 170
- tile_improvement_sprite_static
 - TileImprovement, 170
- tile_improvement_string
 - TileImprovement, 170
- tile_improvement_type
 - TileImprovement, 170
- tile_position_x_vec
 - HexMap, 87
- tile_position_y_vec
 - HexMap, 87
- tile_resource
 - HexTile, 131
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES
 - constants.h, 196
- tile_selected
 - HexMap, 87
- TILE_SELECTED_CHANNEL
 - constants.h, 197
- tile_sprite
 - HexTile, 131
- TILE_STATE_CHANNEL
 - constants.h, 197
- tile_type
 - HexTile, 131
- TILE_TYPE_CUMULATIVE_PROBABILITIES
 - constants.h, 197
- TileImprovement, 158
 - __closeProductionMenu, 162
 - __handleKeyPressEvents, 163
 - __handleMouseButtonEvents, 163
 - __openProductionMenu, 163
 - __setUpProductionMenu, 164
 - ~TileImprovement, 162
 - assets_manager_ptr, 167
 - credits, 167
 - draw, 164
 - event_ptr, 167
 - frame, 168
 - game_phase, 168
 - getTileOptionsSubstring, 166
 - health, 168
 - is_running, 168
 - is_selected, 168
 - just_built, 168
 - message_hub_ptr, 169
 - position_x, 169
 - position_y, 169
 - processEvent, 166
 - processMessage, 166
 - production_menu_backing, 169
 - production_menu_backing_text, 169
 - production_menu_open, 169
 - render_window_ptr, 170
 - setIsSelected, 166
 - tile_improvement_sprite_animated, 170
 - tile_improvement_sprite_static, 170
 - tile_improvement_string, 170
 - tile_improvement_type, 170
 - TileImprovement, 161
- TileImprovement.h
 - DIESEL_GENERATOR, 215
 - ENERGY_STORAGE_SYSTEM, 215
 - N_TILE_IMPROVEMENT_TYPES, 215
 - SETTLEMENT, 215
 - SOLAR_PV, 215
 - TIDAL_TURBINE, 215
 - TileImprovementType, 215
 - WAVE_ENERGY_CONVERTER, 215

- WIND_TURBINE, 215
- TileImprovementType
 - TileImprovement.h, 215
- TileResource
 - HexTile.h, 210
- TileType
 - HexTile.h, 210
- time_since_start_s
 - Game, 65
- toggleResourceOverlay
 - HexMap, 84
 - HexTile, 125
- track_map
 - AssetsManager, 19
- turn
 - Game, 65
- VICTORY
 - Game.h, 208
- visual_screen
 - ContextMenu, 36
- visual_screen_frame_bottom
 - ContextMenu, 36
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, 192
- visual_screen_frame_left
 - ContextMenu, 36
- visual_screen_frame_right
 - ContextMenu, 37
- visual_screen_frame_top
 - ContextMenu, 37
- WAVE_ENERGY_CONVERTER
 - TileImprovement.h, 215
- WAVE_ENERGY_CONVERTER_BUILD_COST
 - constants.h, 197
- WaveEnergyConverter, 171
 - __handleKeyPressEvents, 173
 - __handleMouseButtonEvents, 173
 - __setUpTileImprovementSpriteAnimated, 174
 - ~WaveEnergyConverter, 173
 - draw, 174
 - getTileOptionsSubstring, 175
 - processEvent, 175
 - processMessage, 176
 - WaveEnergyConverter, 172
- WIND_TURBINE
 - TileImprovement.h, 215
- WIND_TURBINE_BUILD_COST
 - constants.h, 197
- WIND_TURBINE_WATER_BUILD_MULTIPLIER
 - constants.h, 198
- WindTurbine, 176
 - __handleKeyPressEvents, 179
 - __handleMouseButtonEvents, 179
 - __setUpTileImprovementSpriteAnimated, 179
 - ~WindTurbine, 178
 - draw, 180
 - getTileOptionsSubstring, 180
 - processEvent, 181
 - processMessage, 181
 - WindTurbine, 178
- year
 - Game, 65