

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 __upgrade()	42
4.3.3.5 draw()	42
4.3.3.6 getTileOptionsSubstring()	44
4.3.3.7 processEvent()	44
4.3.3.8 processMessage()	45
4.3.4 Member Data Documentation	45
4.3.4.1 capacity_kW	45
4.3.4.2 max_production_MWh	45
4.3.4.3 production_MWh	45
4.3.4.4 smoke_da	45
4.3.4.5 smoke_dx	46
4.3.4.6 smoke_dy	46
4.3.4.7 smoke_prob	46
4.3.4.8 smoke_sprite_list	46
4.4 EnergyStorageSystem Class Reference	47
4.4.1 Detailed Description	48
4.4.2 Constructor & Destructor Documentation	48
4.4.2.1 EnergyStorageSystem()	48
4.4.2.2 ~EnergyStorageSystem()	49
4.4.3 Member Function Documentation	49
4.4.3.1 __handleKeyPressEvents()	50
4.4.3.2 __handleMouseButtonEvents()	50
4.4.3.3 __setUpProductionMenu()	51
4.4.3.4 __setUpTileImprovementSpriteStatic()	51
4.4.3.5 __upgrade()	51
4.4.3.6 draw()	52
4.4.3.7 getTileOptionsSubstring()	52
4.4.3.8 processEvent()	53
4.4.3.9 processMessage()	54
4.4.3.10 setIsSelected()	54
4.4.4 Member Data Documentation	54

4.4.4.1 capacity_MWh	54
4.4.4.2 charge_MWh	55
4.5 Game Class Reference	55
4.5.1 Detailed Description	57
4.5.2 Constructor & Destructor Documentation	57
4.5.2.1 Game()	57
4.5.2.2 ~Game()	58
4.5.3 Member Function Documentation	58
4.5.3.1 __draw()	58
4.5.3.2 __drawFrameClockOverlay()	59
4.5.3.3 __drawHUD()	59
4.5.3.4 __handleKeyPressEvents()	61
4.5.3.5 __handleMouseButtonEvents()	61
4.5.3.6 __insufficientCreditsAlarm()	62
4.5.3.7 __processEvent()	63
4.5.3.8 __processMessage()	63
4.5.3.9 __sendGameStateMessage()	64
4.5.3.10 __toggleFrameClockOverlay()	65
4.5.3.11 run()	66
4.5.4 Member Data Documentation	66
4.5.4.1 assets_manager_ptr	67
4.5.4.2 clock	67
4.5.4.3 context_menu_ptr	67
4.5.4.4 credits	67
4.5.4.5 cumulative_emissions_tonnes	67
4.5.4.6 demand_MWh	67
4.5.4.7 event	68
4.5.4.8 frame	68
4.5.4.9 game_loop_broken	68
4.5.4.10 game_phase	68
4.5.4.11 hex_map_ptr	68
4.5.4.12 message_hub	68
4.5.4.13 month	69
4.5.4.14 population	69
4.5.4.15 quit_game	69
4.5.4.16 render_window_ptr	69
4.5.4.17 show_frame_clock_overlay	69
4.5.4.18 time_since_start_s	69
4.5.4.19 turn	70
4.5.4.20 year	70
4.6 HexMap Class Reference	70
4.6.1 Detailed Description	73

4.6.2 Constructor & Destructor Documentation	73
4.6.2.1 HexMap()	73
4.6.2.2 ~HexMap()	74
4.6.3 Member Function Documentation	74
4.6.3.1 __assembleHexMap()	74
4.6.3.2 __assessNeighbours()	74
4.6.3.3 __buildDrawOrderVector()	75
4.6.3.4 __enforceOceanContinuity()	76
4.6.3.5 __getMajorityTileType()	76
4.6.3.6 __getNeighboursVector()	77
4.6.3.7 __getNoise()	78
4.6.3.8 __getSelectedTile()	79
4.6.3.9 __getValidMapIndexPositions()	80
4.6.3.10 __handleKeyPressEvents()	81
4.6.3.11 __handleMouseButtonEvents()	81
4.6.3.12 __isLakeTouchingOcean()	82
4.6.3.13 __layTiles()	82
4.6.3.14 __procedurallyGenerateTileResources()	84
4.6.3.15 __procedurallyGenerateTileTypes()	85
4.6.3.16 __sendNoTileSelectedMessage()	86
4.6.3.17 __setUpGlassScreen()	86
4.6.3.18 __smoothTileTypes()	86
4.6.3.19 assess()	87
4.6.3.20 clear()	87
4.6.3.21 draw()	87
4.6.3.22 processEvent()	88
4.6.3.23 processMessage()	89
4.6.3.24 reroll()	89
4.6.3.25 toggleResourceOverlay()	89
4.6.4 Member Data Documentation	90
4.6.4.1 assets_manager_ptr	90
4.6.4.2 border_tiles_vec	90
4.6.4.3 event_ptr	90
4.6.4.4 frame	90
4.6.4.5 glass_screen	91
4.6.4.6 hex_draw_order_vec	91
4.6.4.7 hex_map	91
4.6.4.8 message_hub_ptr	91
4.6.4.9 n_layers	91
4.6.4.10 n_tiles	91
4.6.4.11 position_x	92
4.6.4.12 position_y	92

4.6.4.13 render_window_ptr	92
4.6.4.14 show_resource	92
4.6.4.15 tile_position_x_vec	92
4.6.4.16 tile_position_y_vec	92
4.6.4.17 tile_selected	93
4.7 HexTile Class Reference	93
4.7.1 Detailed Description	97
4.7.2 Constructor & Destructor Documentation	97
4.7.2.1 HexTile()	97
4.7.2.2 ~HexTile()	98
4.7.3 Member Function Documentation	99
4.7.3.1 __buildDieselGenerator()	99
4.7.3.2 __buildEnergyStorage()	99
4.7.3.3 __buildSettlement()	100
4.7.3.4 __buildSolarPV()	100
4.7.3.5 __buildTidalTurbine()	101
4.7.3.6 __buildWaveEnergyConverter()	102
4.7.3.7 __buildWindTurbine()	102
4.7.3.8 __clearDecoration()	103
4.7.3.9 __closeBuildMenu()	103
4.7.3.10 __getTileCoordsSubstring()	104
4.7.3.11 __getTileImprovementSubstring()	104
4.7.3.12 __getTileOptionsSubstring()	104
4.7.3.13 __getTileResourceSubstring()	106
4.7.3.14 __getTileTypeSubstring()	107
4.7.3.15 __handleKeyPressEvents()	107
4.7.3.16 __handleKeyReleaseEvents()	111
4.7.3.17 __handleMouseButtonEvents()	112
4.7.3.18 __isClicked()	113
4.7.3.19 __openBuildMenu()	113
4.7.3.20 __scrapImprovement()	113
4.7.3.21 __sendAssessNeighboursMessage()	114
4.7.3.22 __sendCreditsSpentMessage()	115
4.7.3.23 __sendGameStateRequest()	115
4.7.3.24 __sendInsufficientCreditsMessage()	115
4.7.3.25 __sendTileSelectedMessage()	116
4.7.3.26 __sendTileStateMessage()	116
4.7.3.27 __sendUpdateGamePhaseMessage()	116
4.7.3.28 __setIsSelected()	117
4.7.3.29 __setResourceText()	117
4.7.3.30 __setUpBuildMenu()	118
4.7.3.31 __setUpBuildOption()	119

4.7.3.32 __setUpDieselGeneratorBuildOption()	120
4.7.3.33 __setUpEnergyStorageSystemBuildOption()	121
4.7.3.34 __setUpMagnifyingGlassSprite()	121
4.7.3.35 __setUpNodeSprite()	122
4.7.3.36 __setUpResourceChipSprite()	122
4.7.3.37 __setUpSelectOutlineSprite()	122
4.7.3.38 __setUpSolarPVBuildOption()	123
4.7.3.39 __setUpTidalTurbineBuildOption()	123
4.7.3.40 __setUpTileExplosionReel()	124
4.7.3.41 __setUpTileSprite()	124
4.7.3.42 __setUpWaveEnergyConverterBuildOption()	124
4.7.3.43 __setUpWindTurbineBuildOption()	125
4.7.3.44 assess()	126
4.7.3.45 decorateTile()	126
4.7.3.46 draw()	127
4.7.3.47 processEvent()	128
4.7.3.48 processMessage()	129
4.7.3.49 setTileResource() [1/2]	129
4.7.3.50 setTileResource() [2/2]	130
4.7.3.51 setTileType() [1/2]	130
4.7.3.52 setTileType() [2/2]	131
4.7.3.53 toggleResourceOverlay()	132
4.7.4 Member Data Documentation	132
4.7.4.1 assets_manager_ptr	132
4.7.4.2 build_menu_backing	132
4.7.4.3 build_menu_backing_text	133
4.7.4.4 build_menu_open	133
4.7.4.5 build_menu_options_text_vec	133
4.7.4.6 build_menu_options_vec	133
4.7.4.7 credits	133
4.7.4.8 decoration_cleared	133
4.7.4.9 draw_explosion	134
4.7.4.10 event_ptr	134
4.7.4.11 explosion_frame	134
4.7.4.12 explosion_sprite_reel	134
4.7.4.13 frame	134
4.7.4.14 game_phase	134
4.7.4.15 has_improvement	135
4.7.4.16 is_selected	135
4.7.4.17 magnifying_glass_sprite	135
4.7.4.18 major_radius	135
4.7.4.19 message_hub_ptr	135

4.7.4.20	minor_radius	135
4.7.4.21	node_sprite	136
4.7.4.22	position_x	136
4.7.4.23	position_y	136
4.7.4.24	render_window_ptr	136
4.7.4.25	resource_assessed	136
4.7.4.26	resource_assessment	136
4.7.4.27	resource_chip_sprite	137
4.7.4.28	resource_text	137
4.7.4.29	scrap_improvement_frame	137
4.7.4.30	select_outline_sprite	137
4.7.4.31	show_node	137
4.7.4.32	show_resource	137
4.7.4.33	tile_decoration_sprite	138
4.7.4.34	tile_improvement_ptr	138
4.7.4.35	tile_resource	138
4.7.4.36	tile_sprite	138
4.7.4.37	tile_type	138
4.8	Message Struct Reference	138
4.8.1	Detailed Description	139
4.8.2	Member Data Documentation	139
4.8.2.1	bool_payload	139
4.8.2.2	channel	139
4.8.2.3	double_payload	139
4.8.2.4	int_payload	140
4.8.2.5	string_payload	140
4.8.2.6	subject	140
4.9	MessageHub Class Reference	140
4.9.1	Detailed Description	141
4.9.2	Constructor & Destructor Documentation	141
4.9.2.1	MessageHub()	141
4.9.2.2	~MessageHub()	141
4.9.3	Member Function Documentation	141
4.9.3.1	addChannel()	141
4.9.3.2	clear()	142
4.9.3.3	clearMessages()	142
4.9.3.4	hasTraffic()	143
4.9.3.5	isEmpty()	143
4.9.3.6	popMessage()	143
4.9.3.7	receiveMessage()	144
4.9.3.8	removeChannel()	145
4.9.3.9	sendMessage()	145

4.9.4 Member Data Documentation	146
4.9.4.1 message_map	146
4.10 Settlement Class Reference	146
4.10.1 Detailed Description	148
4.10.2 Constructor & Destructor Documentation	148
4.10.2.1 Settlement()	148
4.10.2.2 ~Settlement()	149
4.10.3 Member Function Documentation	149
4.10.3.1 __handleKeyPressEvents()	149
4.10.3.2 __handleMouseButtonEvents()	150
4.10.3.3 __setUpTileImprovementSpriteStatic()	150
4.10.3.4 draw()	151
4.10.3.5 getTileOptionsSubstring()	152
4.10.3.6 processEvent()	152
4.10.3.7 processMessage()	152
4.10.3.8 setIsSelected()	153
4.10.4 Member Data Documentation	153
4.10.4.1 smoke_da	153
4.10.4.2 smoke_dx	153
4.10.4.3 smoke_dy	153
4.10.4.4 smoke_prob	154
4.10.4.5 smoke_sprite_list	154
4.11 SolarPV Class Reference	154
4.11.1 Detailed Description	156
4.11.2 Constructor & Destructor Documentation	156
4.11.2.1 SolarPV()	156
4.11.2.2 ~SolarPV()	157
4.11.3 Member Function Documentation	157
4.11.3.1 __drawUpgradeOptions()	157
4.11.3.2 __handleKeyPressEvents()	159
4.11.3.3 __handleMouseButtonEvents()	159
4.11.3.4 __setUpTileImprovementSpriteStatic()	160
4.11.3.5 __upgradePowerCapacity()	160
4.11.3.6 draw()	161
4.11.3.7 getTileOptionsSubstring()	162
4.11.3.8 processEvent()	163
4.11.3.9 processMessage()	163
4.11.4 Member Data Documentation	163
4.11.4.1 capacity_kW	163
4.11.4.2 dispatchable_MWh	163
4.11.4.3 production_MWh	164
4.12 TidalTurbine Class Reference	164

4.12.1 Detailed Description	165
4.12.2 Constructor & Destructor Documentation	165
4.12.2.1 TidalTurbine()	166
4.12.2.2 ~TidalTurbine()	166
4.12.3 Member Function Documentation	167
4.12.3.1 __drawUpgradeOptions()	167
4.12.3.2 __handleKeyPressEvents()	168
4.12.3.3 __handleMouseButtonEvents()	169
4.12.3.4 __setUpTileImprovementSpriteAnimated()	169
4.12.3.5 __upgradePowerCapacity()	170
4.12.3.6 draw()	170
4.12.3.7 getTileOptionsSubstring()	172
4.12.3.8 processEvent()	172
4.12.3.9 processMessage()	173
4.12.4 Member Data Documentation	173
4.12.4.1 capacity_kW	173
4.12.4.2 dispatchable_MWh	173
4.12.4.3 production_MWh	173
4.13 TileImprovement Class Reference	174
4.13.1 Detailed Description	177
4.13.2 Constructor & Destructor Documentation	177
4.13.2.1 TileImprovement()	177
4.13.2.2 ~TileImprovement()	178
4.13.3 Member Function Documentation	178
4.13.3.1 __closeProductionMenu()	178
4.13.3.2 __closeUpgradeMenu()	179
4.13.3.3 __handleKeyPressEvents()	179
4.13.3.4 __handleMouseButtonEvents()	179
4.13.3.5 __openProductionMenu()	180
4.13.3.6 __openUpgradeMenu()	180
4.13.3.7 __sendCreditsSpentMessage()	181
4.13.3.8 __sendGameStateRequest()	181
4.13.3.9 __sendInsufficientCreditsMessage()	181
4.13.3.10 __sendTileStateRequest()	182
4.13.3.11 __setUpProductionMenu()	182
4.13.3.12 __setUpUpgradeMenu()	182
4.13.3.13 __upgradeStorageCapacity()	183
4.13.3.14 draw()	184
4.13.3.15 getTileOptionsSubstring()	185
4.13.3.16 processEvent()	186
4.13.3.17 processMessage()	186
4.13.3.18 setIsSelected()	186

4.13.4 Member Data Documentation	187
4.13.4.1 assets_manager_ptr	187
4.13.4.2 credits	187
4.13.4.3 event_ptr	187
4.13.4.4 frame	187
4.13.4.5 game_phase	187
4.13.4.6 health	188
4.13.4.7 is_running	188
4.13.4.8 is_selected	188
4.13.4.9 just_built	188
4.13.4.10 just_upgraded	188
4.13.4.11 message_hub_ptr	188
4.13.4.12 position_x	189
4.13.4.13 position_y	189
4.13.4.14 production_menu_backing	189
4.13.4.15 production_menu_backing_text	189
4.13.4.16 production_menu_open	189
4.13.4.17 render_window_ptr	189
4.13.4.18 storage_level	190
4.13.4.19 storage_upgrade_sprite	190
4.13.4.20 storage_upgrade_sprite_vec	190
4.13.4.21 tile_improvement_sprite_animated	190
4.13.4.22 tile_improvement_sprite_static	190
4.13.4.23 tile_improvement_string	190
4.13.4.24 tile_improvement_type	191
4.13.4.25 upgrade_arrow_sprite	191
4.13.4.26 upgrade_frame	191
4.13.4.27 upgrade_level	191
4.13.4.28 upgrade_menu_backing	191
4.13.4.29 upgrade_menu_backing_text	191
4.13.4.30 upgrade_menu_open	192
4.13.4.31 upgrade_plus_sprite	192
4.14 WaveEnergyConverter Class Reference	192
4.14.1 Detailed Description	194
4.14.2 Constructor & Destructor Documentation	194
4.14.2.1 WaveEnergyConverter()	194
4.14.2.2 ~WaveEnergyConverter()	195
4.14.3 Member Function Documentation	195
4.14.3.1 __drawUpgradeOptions()	195
4.14.3.2 __handleKeyPressEvents()	197
4.14.3.3 __handleMouseButtonEvents()	197
4.14.3.4 __setUpTileImprovementSpriteAnimated()	198

4.14.3.5 __upgradePowerCapacity()	199
4.14.3.6 draw()	199
4.14.3.7 getTileOptionsSubstring()	200
4.14.3.8 processEvent()	201
4.14.3.9 processMessage()	201
4.14.4 Member Data Documentation	202
4.14.4.1 capacity_kW	202
4.14.4.2 dispatchable_MWh	202
4.14.4.3 production_MWh	202
4.15 WindTurbine Class Reference	202
4.15.1 Detailed Description	204
4.15.2 Constructor & Destructor Documentation	204
4.15.2.1 WindTurbine()	204
4.15.2.2 ~WindTurbine()	205
4.15.3 Member Function Documentation	205
4.15.3.1 __drawUpgradeOptions()	205
4.15.3.2 __handleKeyPressEvents()	207
4.15.3.3 __handleMouseButtonEvents()	207
4.15.3.4 __setUpTileImprovementSpriteAnimated()	208
4.15.3.5 __upgradePowerCapacity()	209
4.15.3.6 draw()	209
4.15.3.7 getTileOptionsSubstring()	210
4.15.3.8 processEvent()	211
4.15.3.9 processMessage()	211
4.15.4 Member Data Documentation	212
4.15.4.1 capacity_kW	212
4.15.4.2 dispatchable_MWh	212
4.15.4.3 production_MWh	212
5 File Documentation	213
5.1 header/ContextMenu.h File Reference	213
5.1.1 Detailed Description	214
5.1.2 Enumeration Type Documentation	214
5.1.2.1 ConsoleState	214
5.2 header/DieselGenerator.h File Reference	214
5.2.1 Detailed Description	215
5.3 header/EnergyStorageSystem.h File Reference	215
5.3.1 Detailed Description	216
5.4 header/ESC_core/AssetsManager.h File Reference	216
5.4.1 Detailed Description	217
5.5 header/ESC_core/constants.h File Reference	217
5.5.1 Detailed Description	219

5.5.2 Function Documentation	220
5.5.2.1 FOREST_GREEN()	220
5.5.2.2 LAKE_BLUE()	220
5.5.2.3 MENU_FRAME_GREY()	220
5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()	220
5.5.2.5 MONOCHROME_TEXT_AMBER()	221
5.5.2.6 MONOCHROME_TEXT_GREEN()	221
5.5.2.7 MONOCHROME_TEXT_RED()	221
5.5.2.8 MOUNTAINS_GREY()	221
5.5.2.9 OCEAN_BLUE()	221
5.5.2.10 PLAINS_YELLOW()	222
5.5.2.11 RESOURCE_CHIP_GREY()	222
5.5.2.12 VISUAL_SCREEN_FRAME_GREY()	222
5.5.3 Variable Documentation	222
5.5.3.1 BUILD_SETTLEMENT_COST	222
5.5.3.2 CLEAR_FOREST_COST	222
5.5.3.3 CLEAR_MOUNTAINS_COST	223
5.5.3.4 CLEAR_PLAINS_COST	223
5.5.3.5 CO2E_KG_PER_LITRE_DIESEL	223
5.5.3.6 DIESEL_GENERATOR_BUILD_COST	223
5.5.3.7 EMISSIONS_LIFETIME_LIMIT_TONNES	223
5.5.3.8 ENERGY_STORAGE_SYSTEM_BUILD_COST	223
5.5.3.9 FLOAT_TOLERANCE	224
5.5.3.10 FRAMES_PER_SECOND	224
5.5.3.11 GAME_CHANNEL	224
5.5.3.12 GAME_HEIGHT	224
5.5.3.13 GAME_STATE_CHANNEL	224
5.5.3.14 GAME_WIDTH	224
5.5.3.15 HEX_MAP_CHANNEL	225
5.5.3.16 MAX_STORAGE_LEVELS	225
5.5.3.17 MAX_UPGRADE_LEVELS	225
5.5.3.18 NO_TILE_SELECTED_CHANNEL	225
5.5.3.19 RESOURCE_ASSESSMENT_COST	225
5.5.3.20 SCRAP_COST	225
5.5.3.21 SECONDS_PER_FRAME	226
5.5.3.22 SECONDS_PER_MONTH	226
5.5.3.23 SECONDS_PER_YEAR	226
5.5.3.24 SOLAR_PV_BUILD_COST	226
5.5.3.25 SOLAR_PV_WATER_BUILD_MULTIPLIER	226
5.5.3.26 STARTING_CREDITS	226
5.5.3.27 STARTING_POPULATION	227
5.5.3.28 TIDAL_TURBINE_BUILD_COST	227

5.5.3.29 TILE_RESOURCE_CUMULATIVE_PROBABILITIES	227
5.5.3.30 TILE_SELECTED_CHANNEL	227
5.5.3.31 TILE_STATE_CHANNEL	227
5.5.3.32 TILE_TYPE_CUMULATIVE_PROBABILITIES	228
5.5.3.33 WAVE_ENERGY_CONVERTER_BUILD_COST	228
5.5.3.34 WIND_TURBINE_BUILD_COST	228
5.5.3.35 WIND_TURBINE_WATER_BUILD_MULTIPLIER	228
5.6 header/ESC_core/doxygen_cite.h File Reference	228
5.6.1 Detailed Description	228
5.7 header/ESC_core/includes.h File Reference	229
5.7.1 Detailed Description	229
5.8 header/ESC_core/MessageHub.h File Reference	230
5.8.1 Detailed Description	230
5.9 header/ESC_core/testing_utils.h File Reference	230
5.9.1 Detailed Description	231
5.9.2 Function Documentation	231
5.9.2.1 expectedErrorNotDetected()	232
5.9.2.2 printGold()	232
5.9.2.3 printGreen()	232
5.9.2.4 printRed()	233
5.9.2.5 testFloatEquals()	233
5.9.2.6 testGreaterThan()	234
5.9.2.7 testGreaterThanOrEqualTo()	234
5.9.2.8 testLessThan()	235
5.9.2.9 testLessThanOrEqualTo()	236
5.9.2.10 testTruth()	236
5.10 header/Game.h File Reference	237
5.10.1 Enumeration Type Documentation	238
5.10.1.1 GamePhase	238
5.11 header/HexMap.h File Reference	238
5.11.1 Detailed Description	239
5.12 header/HexTile.h File Reference	239
5.12.1 Detailed Description	240
5.12.2 Enumeration Type Documentation	240
5.12.2.1 TileResource	240
5.12.2.2 TileType	241
5.13 header/Settlement.h File Reference	241
5.13.1 Detailed Description	242
5.14 header/SolarPV.h File Reference	242
5.14.1 Detailed Description	243
5.15 header/TidalTurbine.h File Reference	243
5.15.1 Detailed Description	244

5.16 header/TileImprovement.h File Reference	244
5.16.1 Detailed Description	245
5.16.2 Enumeration Type Documentation	245
5.16.2.1 TileImprovementType	245
5.17 header/WaveEnergyConverter.h File Reference	246
5.17.1 Detailed Description	247
5.18 header/WindTurbine.h File Reference	247
5.18.1 Detailed Description	248
5.19 source/ContextMenu.cpp File Reference	248
5.19.1 Detailed Description	248
5.20 source/DieselGenerator.cpp File Reference	249
5.20.1 Detailed Description	249
5.21 source/EnergyStorageSystem.cpp File Reference	249
5.21.1 Detailed Description	249
5.22 source/ESC_core/AssetsManager.cpp File Reference	249
5.22.1 Detailed Description	250
5.23 source/ESC_core/MessageHub.cpp File Reference	250
5.23.1 Detailed Description	250
5.24 source/ESC_core/testing_utils.cpp File Reference	250
5.24.1 Detailed Description	251
5.24.2 Function Documentation	251
5.24.2.1 expectedErrorNotDetected()	251
5.24.2.2 printGold()	252
5.24.2.3 printGreen()	252
5.24.2.4 printRed()	252
5.24.2.5 testFloatEquals()	253
5.24.2.6 testGreaterThan()	253
5.24.2.7 testGreaterThanOrEqualTo()	254
5.24.2.8 testLessThan()	255
5.24.2.9 testLessThanOrEqualTo()	255
5.24.2.10 testTruth()	256
5.25 source/Game.cpp File Reference	256
5.25.1 Detailed Description	257
5.26 source/HexMap.cpp File Reference	257
5.26.1 Detailed Description	257
5.27 source/HexTile.cpp File Reference	257
5.27.1 Detailed Description	258
5.28 source/main.cpp File Reference	258
5.28.1 Detailed Description	258
5.28.2 Function Documentation	258
5.28.2.1 constructRenderWindow()	258
5.28.2.2 loadAssets()	259

5.28.2.3 main()	261
5.29 source/Settlement.cpp File Reference	262
5.29.1 Detailed Description	262
5.30 source/SolarPV.cpp File Reference	262
5.30.1 Detailed Description	263
5.31 source/TidalTurbine.cpp File Reference	263
5.31.1 Detailed Description	263
5.32 source/TileImprovement.cpp File Reference	263
5.32.1 Detailed Description	263
5.33 source/WaveEnergyConverter.cpp File Reference	264
5.33.1 Detailed Description	264
5.34 source/WindTurbine.cpp File Reference	264
5.34.1 Detailed Description	264
Bibliography	265
Index	267

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AssetsManager	7
ContextMenu	19
Game	55
HexMap	70
HexTile	93
Message	138
MessageHub	140
TileImprovement	174
DieselGenerator	37
EnergyStorageSystem	47
Settlement	146
SolarPV	154
TidalTurbine	164
WaveEnergyConverter	192
WindTurbine	202

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)	47
Game	A class which acts as the central class for the game, by containing all other classes and implementing the game loop	55
HexMap	A class which defines a hex map of hex tiles	70
HexTile	A class which defines a hex tile of the hex map	93
Message	A structure which defines a standard message format	138
MessageHub	A class which acts as a central hub for inter-object message traffic	140
Settlement	A settlement class (child class of TileImprovement)	146
SolarPV	A settlement class (child class of TileImprovement)	154
TidalTurbine	A settlement class (child class of TileImprovement)	164
TileImprovement	A base class for the tile improvement hierarchy	174
WaveEnergyConverter	A settlement class (child class of TileImprovement)	192
WindTurbine	A settlement class (child class of TileImprovement)	202

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	213
header/ DieselGenerator.h	
Header file for the DieselGenerator class	214
header/ EnergyStorageSystem.h	
Header file for the EnergyStorageSystem class	215
header/ Game.h	237
header/ HexMap.h	
Header file for the HexMap class	238
header/ HexTile.h	
Header file for the Game class	239
header/ Settlement.h	
Header file for the Settlement class	241
header/ SolarPV.h	
Header file for the SolarPV class	242
header/ TidalTurbine.h	
Header file for the TidalTurbine class	243
header/ TileImprovement.h	
Header file for the TileImprovement class	244
header/ WaveEnergyConverter.h	
Header file for the WaveEnergyConverter class	246
header/ WindTurbine.h	
Header file for the WindTurbine class	247
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	216
header/ESC_core/ constants.h	
Header file for various constants	217
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	228
header/ESC_core/ includes.h	
Header file for various includes	229
header/ESC_core/ MessageHub.h	
Header file for the MessageHub class	230
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	230

source/ ContextMenu.cpp	Implementation file for the ContextMenu class	248
source/ DieselGenerator.cpp	Implementation file for the DieselGenerator class	249
source/ EnergyStorageSystem.cpp	Implementation file for the EnergyStorageSystem class	249
source/ Game.cpp	Implementation file for the Game class	256
source/ HexMap.cpp	Implementation file for the HexMap class	257
source/ HexTile.cpp	Implementation file for the HexTile class	257
source/ main.cpp	Implementation file for main() for Road To Zero	258
source/ Settlement.cpp	Implementation file for the Settlement class	262
source/ SolarPV.cpp	Implementation file for the SolarPV class	262
source/ TidalTurbine.cpp	Implementation file for the TidalTurbine class	263
source/ TileImprovement.cpp	Implementation file for the TileImprovement class	263
source/ WaveEnergyConverter.cpp	Implementation file for the WaveEnergyConverter class	264
source/ WindTurbine.cpp	Implementation file for the WindTurbine class	264
source/ESC_core/ AssetsManager.cpp	Implementation file for the AssetsManager class	249
source/ESC_core/ MessageHub.cpp	Implementation file for the MessageHub class	250
source/ESC_core/ testing_utils.cpp	Implementation file for various testing utilities	250

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:

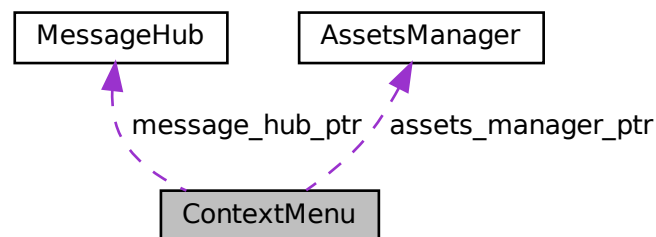
- header/ESC_core/[AssetsManager.h](#)
- source/ESC_core/[AssetsManager.cpp](#)

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(MONOCHROME_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         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:

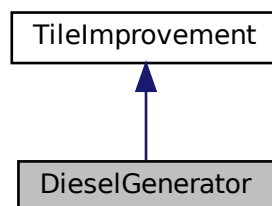
- header/[ContextMenu.h](#)
- source/[ContextMenu.cpp](#)

4.3 DieselGenerator Class Reference

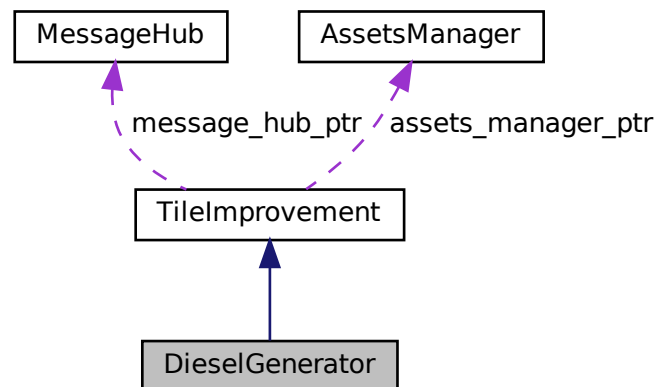
A settlement class (child class of [TileImprovement](#)).

```
#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 [__upgrade](#) (void)
Helper method to upgrade the diesel generator.
- 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.

```
275 :
276 TileImprovement (
277     position_x,
278     position_y,
279     event_ptr,
280     render_window_ptr,
```

```

281     assets_manager_ptr,
282     message_hub_ptr
283 )
284 {
285     // 1. set attributes
286
287     // 1.1. private
288     //...
289
290     // 1.2. public
291     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
292
293     this->is_running = false;
294
295     this->health = 100;
296
297     this->capacity_kW = 100;
298     this->upgrade_level = 1;
299
300     this->production_MWh = 0;
301     this->max_production_MWh = 72;
302
303     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
304     this->smoke_dx = 5 * SECONDS_PER_FRAME;
305     this->smoke_dy = -10 * SECONDS_PER_FRAME;
306     this->smoke_prob = 8 * SECONDS_PER_FRAME;
307
308     this->smoke_sprite_list = {};
309
310     this->tile_improvement_string = "DIESEL GEN";
311
312     this->__setUpTileImprovementSpriteAnimated();
313
314     std::cout << "DieselGenerator constructed at " << this << std::endl;
315
316     return;
317 } /* DieselGenerator() */

```

4.3.2.2 ~DieselGenerator()

```

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

```

Destructor for the [DieselGenerator](#) class.

```

526 {
527     std::cout << "DieselGenerator at " << this << " destroyed" << std::endl;
528
529     return;
530 } /* ~DieselGenerator() */

```

4.3.3 Member Function Documentation

4.3.3.1 __handleKeyPressEvents()

```

void DieselGenerator::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

161 {
162     if (this->just_built) {
163         return;
164     }
165
166
167     switch (this->event_ptr->key.code) {

```

```

168         case (sf::Keyboard::U): {
169             this->__upgrade();
170
171             break;
172         }
173
174         default: {
175             // do nothing!
176
177             break;
178         }
179     }
180 }
181
182
183 return;
184 } /* __handleKeyPressEvents() */

```

4.3.3.2 __handleMouseButtonEvents()

```

void DieselGenerator::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

199 {
200     if (this->just_built) {
201         return;
202     }
203
204     switch (this->event_ptr->mouseButton.button) {
205         case (sf::Mouse::Left): {
206             //...
207
208             break;
209         }
210
211         case (sf::Mouse::Right): {
212             //...
213
214             break;
215         }
216
217         default: {
218             // do nothing!
219
220             break;
221         }
222     }
223 }
224
225
226 return;
227 } /* __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 __upgrade()

```

void DieselGenerator::__upgrade (
    void ) [private]

```

Helper method to upgrade the diesel generator.

```

114 {
115     if (this->credits < DIESEL_GENERATOR_BUILD_COST) {
116         std::cout << "Cannot upgrade diesel generator: insufficient credits (need "
117             << DIESEL_GENERATOR_BUILD_COST << " K)" << std::endl;
118
119         this->__sendInsufficientCreditsMessage();
120         return;
121     }
122
123     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
124         return;
125     }
126
127     this->is_running = false;
128
129     this->health = 100;
130
131     this->capacity_kW += 100;
132     this->upgrade_level++;
133
134     this->production_MWh = 0;
135     this->max_production_MWh += 72;
136
137     this->just_upgraded = true;
138
139     this->assets_manager_ptr->getSound("upgrade")->play();
140
141     this->__sendCreditsSpentMessage(DIESEL_GENERATOR_BUILD_COST);
142     this->__sendTileStateRequest();
143     this->__sendGameStateRequest();
144
145     return;
146 } /* __upgrade() */

```

4.3.3.5 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](#).

```

434 {
435     // 1. if just built, call base method and return
436     if (this->just_built) {
437         TileImprovement::draw();
438
439         return;
440     }
441
442     // 2. handle upgrade effects
443     if (this->just_upgraded) {
444         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
445             this->tile_improvement_sprite_animated[i].setColor(
446                 sf::Color(
447                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
448                     255,
449                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
450                     255
451                 )
452             );
453
454             this->tile_improvement_sprite_animated[i].setScale(
455                 sf::Vector2f(
456                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
457                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
458                 )
459             );
460         }
461
462         this->upgrade_frame++;
463     }
464
465     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
466         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
467             this->tile_improvement_sprite_animated[i].setColor(
468                 sf::Color(255,255,255,255)
469             );
470
471             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
472         }
473
474         this->just_upgraded = false;
475         this->upgrade_frame = 0;
476     }
477
478     // 3. draw first element of animated sprite
479     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
480
481     // 4. draw second element of animated sprite
482     if (this->is_running) {
483         //...
484     }
485
486     else {
487         //...
488     }
489
490     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
491
492     // 5. draw smoke effects
493     if (this->is_running) {
494         //...
495     }
496
497     // 6. draw production menu
498     if (this->production_menu_open) {
499         this->render_window_ptr->draw(this->production_menu_backing);
500         this->render_window_ptr->draw(this->production_menu_backing_text);
501
502         //...
503     }
504
505     this->frame++;
506     return;
507 }
508
509 /* draw() */
510 }

```

4.3.3.6 getFileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
334 {
335     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
336
337     //          32 char x 17 line console "-----\n";
338     std::string options_substring = "CAPACITY: ";
339     options_substring += std::to_string(this->capacity_kW);
340     options_substring += " kW (level ";
341     options_substring += std::to_string(this->upgrade_level);
342     options_substring += ") \n";
343
344     options_substring += "PRODUCTION: ";
345     options_substring += std::to_string(this->production_MWh);
346     options_substring += " MWh (MAX ";
347     options_substring += std::to_string(this->max_production_MWh);
348     options_substring += ") \n";
349
350     options_substring += "HEALTH: ";
351     options_substring += std::to_string(this->health);
352     options_substring += "/100 \n";
353
354     options_substring += " \n";
355     options_substring += " **** DIESEL GEN OPTIONS **** \n";
356     options_substring += " \n";
357     options_substring += " [E]: OPEN PRODUCTION MENU \n";
358
359     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
360         options_substring += " [U]: + 100 kW (";
361         options_substring += std::to_string(upgrade_cost);
362         options_substring += " K) \n";
363     }
364
365     options_substring += "HOLD [P]: SCRAP (";
366     options_substring += std::to_string(SCRAP_COST);
367     options_substring += " K)";
368
369     return options_substring;
370 } /* getFileOptionsSubstring() */
```

4.3.3.7 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
385 {
386     TileImprovement :: processEvent ();
387
388     if (this->event_ptr->type == sf::Event::KeyPressed) {
389         this->__handleKeyPressEvents ();
390     }
391
392     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
393         this->__handleMouseButtonEvents ();
394     }
395
396     return;
397 } /* processEvent() */
```

4.3.3.8 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
412 {
413     TileImprovement :: processMessage ();
414
415     //...
416
417     return;
418 } /* 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:

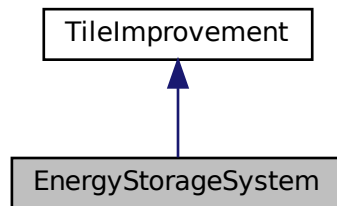
- header/[DieselGenerator.h](#)
- source/[DieselGenerator.cpp](#)

4.4 EnergyStorageSystem Class Reference

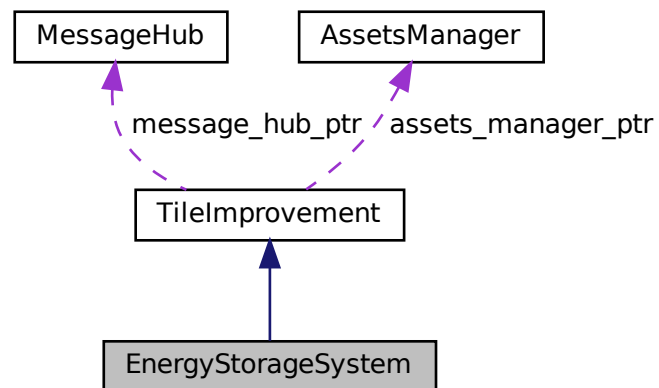
A settlement class (child class of [TileImprovement](#)).

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

Public Attributes

- int [capacity_MWh](#)
The rated energy capacity [MWh] of the energy storage system.
- int [charge_MWh](#)
The charge [MWh] in the energy storage system.

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__setUpProductionMenu](#) (void)
Helper method to set up and position production menu assets (drawable).
- void [__upgrade](#) (void)
Helper method to upgrade the diesel generator.
- 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.

```

291 :
292 TileImprovement (
293     position_x,
294     position_y,
295     event_ptr,
296     render_window_ptr,
297     assets_manager_ptr,
298     message_hub_ptr
299 )
300 {
301     // 1. set attributes
302
303     // 1.1. private
304     //...
305
306     // 1.2. public
307     this->tile_improvement_type = TileImprovementType :: ENERGY_STORAGE_SYSTEM;
308
309     this->is_running = false;
310
311     this->health = 100;
312
313     this->capacity_MWh = 1;
314     this->upgrade_level = 1;
315
316     this->charge_MWh = 0;
317
318     this->tile_improvement_string = "ENERGY STORAGE";
319
320     this->__setUpTileImprovementSpriteStatic();
321     this->__setUpProductionMenu();
322
323     std::cout << "EnergyStorageSystem constructed at " << this << std::endl;
324
325     return;
326 } /* EnergyStorageSystem() */

```

4.4.2.2 ~EnergyStorageSystem()

```

EnergyStorageSystem::~EnergyStorageSystem (
    void ) [virtual]

```

Destructor for the [EnergyStorageSystem](#) class.

```

504 {
505     std::cout << "EnergyStorageSystem at " << this << " destroyed" << std::endl;
506
507     return;
508 } /* ~EnergyStorageSystem() */

```

4.4.3 Member Function Documentation

4.4.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
179 {
180     if (this->just_built) {
181         return;
182     }
183
184     switch (this->event_ptr->key.code) {
185         case (sf::Keyboard::U): {
186             if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
187                 this->__upgrade();
188             }
189
190             break;
191         }
192
193         default: {
194             // do nothing!
195
196             break;
197         }
198     }
199 }
200
201 return;
202 } /* __handleKeyPressEvents() */
```

4.4.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
217 {
218     if (this->just_built) {
219         return;
220     }
221
222     switch (this->event_ptr->mouseButton.button) {
223         case (sf::Mouse::Left): {
224             //...
225
226             break;
227         }
228
229         case (sf::Mouse::Right): {
230             //...
231
232             break;
233         }
234     }
235
236     default: {
237         // do nothing!
238
239         break;
240     }
241 }
242
243 return;
244 } /* __handleMouseButtonEvents() */
```

4.4.3.3 __setUpProductionMenu()

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

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

```
103 {
104     // 1. modify production menu text
105     this->production_menu_backing_text.setString("**** DISCHARGE MENU ****");
106     this->production_menu_backing_text.setFont (
107         *(this->assets_manager_ptr->getFont ("Glass_TTY_VT220"))
108     );
109     this->production_menu_backing_text.setCharacterSize(16);
110     this->production_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
111     this->production_menu_backing_text.setOrigin(
112         this->production_menu_backing_text.getLocalBounds().width / 2, 0
113     );
114     this->production_menu_backing_text.setPosition(400, 400 - 128 + 4);
115
116     return;
117 } /* __setUpProductionMenu() */
```

4.4.3.4 __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
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() */
```

4.4.3.5 __upgrade()

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

Helper method to upgrade the diesel generator.

```
132 {
133     /*
134     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
135
136     if (this->credits < upgrade_cost) {
137         std::cout << "Cannot upgrade diesel generator: insufficient credits (need "
138             << upgrade_cost << " K)" << std::endl;
139
140         this->__sendInsufficientCreditsMessage();
141         return;
142     }
143     */
144 }
```

```

142     }
143
144     this->is_running = false;
145
146     this->health = 100;
147
148     this->capacity_kW += 100;
149     this->upgrade_level++;
150
151     this->production_MWh = 0;
152     this->max_production_MWh += 72;
153
154     this->just_upgraded = true;
155
156     this->assets_manager_ptr->getSound("upgrade")->play();
157
158     this->__sendCreditsSpentMessage(upgrade_cost);
159     this->__sendTileStateRequest();
160     this->__sendGameStateRequest();
161     */
162
163     return;
164 } /* __upgrade() */

```

4.4.3.6 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](#).

```

466 {
467     // 1. if just built, call base method and return
468     if (this->just_built) {
469         TileImprovement :: draw();
470
471         return;
472     }
473
474
475     // 2. draw static sprite
476     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
477
478
479     // 3. draw production menu
480     if (this->production_menu_open) {
481         this->render_window_ptr->draw(this->production_menu_backing);
482         this->render_window_ptr->draw(this->production_menu_backing_text);
483
484         //...
485     }
486
487     this->frame++;
488     return;
489 } /* draw() */

```

4.4.3.7 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

368 {
369     int upgrade_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
370
371     // 32 char x 17 line console "-----\n";
372     std::string options_substring = "CAPACITY: ";
373     options_substring += std::to_string(this->capacity_MWh);
374     options_substring += " MWh (level ";
375     options_substring += std::to_string(this->upgrade_level);
376     options_substring += ")\n";
377
378     options_substring += "CHARGE: ";
379     options_substring += std::to_string(this->charge_MWh);
380     options_substring += " MWh\n";
381
382     options_substring += "HEALTH: ";
383     options_substring += std::to_string(this->health);
384     options_substring += "/100\n";
385
386     options_substring += "\n";
387     options_substring += "**** ENERGY STORAGE OPTIONS ****\n";
388     options_substring += "\n";
389     options_substring += "      [E]:  OPEN DISCHARGE MENU  \n";
390
391     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
392         options_substring += "      [U]:  UPGRADE (";
393         options_substring += std::to_string(upgrade_cost);
394         options_substring += " K)\n";
395     }
396
397     options_substring += "HOLD [P]:  SCRAP (";
398     options_substring += std::to_string(SCRAP_COST);
399     options_substring += " K)";
400
401     return options_substring;
402 } /* getTileOptionsSubstring() */

```

4.4.3.8 processEvent()

```

void EnergyStorageSystem::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

417 {
418     TileImprovement :: processEvent();
419
420     if (this->event_ptr->type == sf::Event::KeyPressed) {
421         this->__handleKeyPressEvents();
422     }
423
424     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
425         this->__handleMouseButtonEvents();
426     }
427
428     return;
429 } /* processEvent() */

```

4.4.3.9 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
444 {
445     TileImprovement :: processMessage();
446
447     //...
448
449     return;
450 } /* processMessage() */
```

4.4.3.10 setIsSelected()

```
void EnergyStorageSystem::setIsSelected (
    bool is_selected ) [virtual]
```

Method to set the is selected attribute.

Parameters

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

Reimplemented from [TileImprovement](#).

```
343 {
344     TileImprovement :: setIsSelected(is_selected);
345
346     if (this->is_selected) {
347         this->assets_manager_ptr->getSound("energy storage system")->play();
348     }
349
350     return;
351 } /* setIsSelected() */
```

4.4.4 Member Data Documentation

4.4.4.1 capacity_MWh

```
int EnergyStorageSystem::capacity_MWh
```

The rated energy capacity [MWh] of the energy storage system.

4.4.4.2 charge_MWh

```
int EnergyStorageSystem::charge_MWh
```

The charge [MWh] in the energy storage system.

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

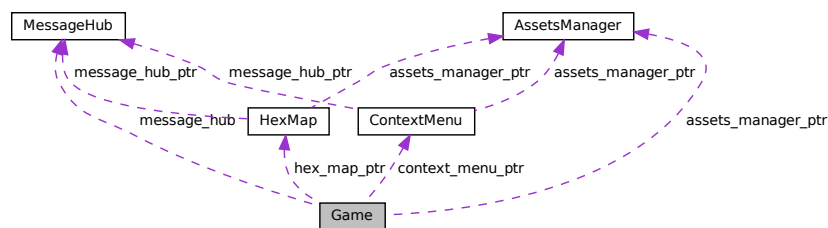
- header/[EnergyStorageSystem.h](#)
- source/[EnergyStorageSystem.cpp](#)

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     case (GamePhase :: SYSTEM_MANAGEMENT): {
599         HUD_string += "SYSTEM MANAGEMENT";
600
601         break;
602     }
603
604     case (GamePhase :: LOSS_EMISSIONS): {
605         HUD_string += "LOSS (EMISSIONS)";
606
607         break;
608     }
609
610     case (GamePhase :: LOSS_DEMAND): {
611         HUD_string += "LOSS (DEMAND)";
612
613         break;
614     }
615
616     case (GamePhase :: LOSS_CREDITS): {
617         HUD_string += "LOSS (CREDITS)";
618
619         break;
620     }
621
622     case (GamePhase :: VICTORY): {
623         HUD_string += "VICTORY";
624
625         break;
626     }
627
628     default: {
629         HUD_string += "???";
630
631         break;
632     }
633 }
634
635 HUD_string += "    TURN: ";
636 HUD_string += std::to_string(this->turn);
637
638 HUD_text.setString(HUD_string);
639
640 HUD_text.setPosition(
641     (800 - HUD_text.getLocalBounds().width) / 2,
642     GAME_HEIGHT - 35
643 );
644
645 this->render_window_ptr->draw(HUD_text);
646
647 return;
648 } /* __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.setFillColor(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(MONOCROME_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 }
```



```

466         this->render_window_ptr->display();
467
468         alarm_frame++;
469         this->frame++;
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
220             break;
221         }
222
223         case (GamePhase :: SYSTEM_MANAGEMENT): {
224             game_state_message.string_payload["game phase"] = "system management";
225
226             break;
227         }
228
229         case (GamePhase :: LOSS_EMISSIONS): {
230             game_state_message.string_payload["game phase"] = "loss emissions";
231
232             break;
233         }
234
235         case (GamePhase :: LOSS_DEMAND): {
236             game_state_message.string_payload["game phase"] = "loss demand";
237
238             break;
239         }
240
241         case (GamePhase :: LOSS_CREDITS): {
242             game_state_message.string_payload["game phase"] = "loss credits";
243
244             break;
245         }
246
247         case (GamePhase :: VICTORY): {
248             game_state_message.string_payload["game phase"] = "victory";
249
250             break;
251         }
252
253         default: {
254             // do nothing!
255
256             break;
257         }
258     }
259
260     this->message_hub.sendMessage(game_state_message);
261
262     std::cout << "Game state message sent by " << this << std::endl;
263     return;
264 } /* __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             // 6.3. draw frame
798             this->render_window_ptr->clear();
799
800             this->hex_map_ptr->draw();
801             this->context_menu_ptr->draw();
802             this->__draw();
803
804             this->render_window_ptr->display();
805
806             // 6.4. increment frame
807             this->frame++;
808         }
809
810         // check track status, move to next if stopped
811         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
812             this->assets_manager_ptr->nextTrack();
813             this->assets_manager_ptr->playTrack();
814         }
815     }
816
817     return this->quit_game;
818 } /* 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:

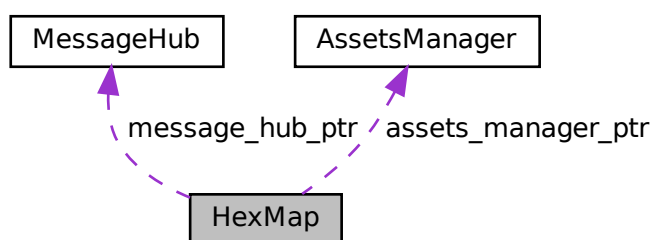
- [header/Game.h](#)
- [source/Game.cpp](#)

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     return;
325 } /* __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

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

Returns

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

```
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:

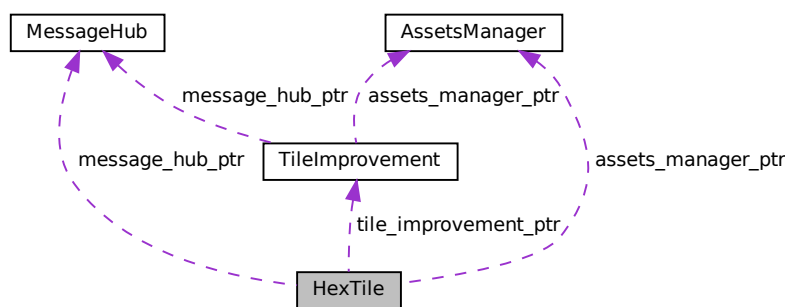
- header/[HexMap.h](#)
- source/[HexMap.cpp](#)

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.
- int [scrap_improvement_frame](#)
A frame for key-hold to confirm scrapping.
- 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` [__setResourceText](#) (`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 [__handleKeyReleaseEvents](#) (void)
- 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). Requires the mapped key to be held continuously to confirm.
- 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.

```

2303 {
2304     // 1. set attributes
2305
2306     // 1.1. private
2307     this->event_ptr = event_ptr;
2308     this->render_window_ptr = render_window_ptr;
2309
2310     this->assets_manager_ptr = assets_manager_ptr;
2311     this->message_hub_ptr = message_hub_ptr;
2312
2313     // 1.2. public
2314     this->show_node = false;
2315     this->show_resource = false;
2316     this->resource_assessed = false;
2317     this->resource_assessment = false;
2318     this->is_selected = false;
2319     this->draw_explosion = false;
2320
2321     this->decoration_cleared = false;
2322     this->has_improvement = false;
2323     this->tile_improvement_ptr = NULL;
2324
2325     this->build_menu_open = false;
2326
2327     this->explosion_frame = 0;
2328
2329     this->frame = 0;
2330     this->credits = 0;
2331
2332     this->scrap_improvement_frame = 0;
2333
2334     this->position_x = position_x;
2335     this->position_y = position_y;
2336
2337     this->major_radius = 32;
2338     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2339
2340     this->game_phase = "build settlement";
2341
2342     // 2. set up and position drawable attributes
2343     this->__setUpNodeSprite();
2344     this->__setUpTileSprite();
2345     this->__setUpSelectOutlineSprite();
2346     this->__setUpResourceChipSprite();
2347     this->__setUpResourceText();
2348     this->__setUpMagnifyingGlassSprite();
2349     this->__setUpTileExplosionReel();
2350
2351     // 3. set tile type and resource (default to none type and average)
2352     this->setTileType(TileType :: NONE_TYPE);
2353     this->setTileResource(TileResource :: AVERAGE);
2354
2355     std::cout << "HexTile constructed at " << this << std::endl;
2356
2357     return;
2358 } /* HexTile() */

```

4.7.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

2913 {
2914     if (this->tile_improvement_ptr != NULL) {
2915         delete this->tile_improvement_ptr;
2916     }
2917
2918     std::cout << "HexTile at " << this << " destroyed" << std::endl;
2919
2920     return;
2921 } /* ~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.

```

1409 {
1410     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1411
1412     if (this->credits < build_cost) {
1413         std::cout << "Cannot build diesel generator: insufficient credits (need "
1414             << build_cost << " K)" << std::endl;
1415
1416         this->__sendInsufficientCreditsMessage();
1417         return;
1418     }
1419
1420     this->tile_improvement_ptr = new DieselGenerator(
1421         this->position_x,
1422         this->position_y,
1423         this->event_ptr,
1424         this->render_window_ptr,
1425         this->assets_manager_ptr,
1426         this->message_hub_ptr
1427     );
1428
1429     this->has_improvement = true;
1430     this->__closeBuildMenu();
1431
1432     this->__sendCreditsSpentMessage(build_cost);
1433     this->__sendTileStateMessage();
1434     this->__sendGameStateRequest();
1435
1436     return;
1437 } /* __buildDieselGenerator() */

```

4.7.3.2 __buildEnergyStorage()

```

void HexTile::__buildEnergyStorage (
    void ) [private]

```

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

```

1652 {
1653     /*
1654     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1655
1656     if (this->credits < build_cost) {
1657         std::cout << "Cannot build energy storage system: insufficient credits (need "
1658             << build_cost << " K)" << std::endl;
1659
1660         this->__sendInsufficientCreditsMessage();
1661         return;
1662     }

```

```

1663
1664     this->tile_improvement_ptr = new EnergyStorageSystem(
1665         this->position_x,
1666         this->position_y,
1667         this->event_ptr,
1668         this->render_window_ptr,
1669         this->assets_manager_ptr,
1670         this->message_hub_ptr
1671     );
1672
1673     this->has_improvement = true;
1674     this->__closeBuildMenu();
1675
1676     this->__sendCreditsSpentMessage(build_cost);
1677     this->__sendTileStateMessage();
1678     this->__sendGameStateRequest();
1679     */
1680     return;
1681 } /* __buildEnergyStorage() */

```

4.7.3.3 __buildSettlement()

```

void HexTile::__buildSettlement (
    void ) [private]

```

Helper method to build a settlement on this tile.

```

1363 {
1364     if (this->credits < BUILD_SETTLEMENT_COST) {
1365         std::cout << "Cannot build settlement: insufficient credits (need "
1366             << BUILD_SETTLEMENT_COST << " K)" << std::endl;
1367
1368         this->__sendInsufficientCreditsMessage();
1369         return;
1370     }
1371
1372     this->__clearDecoration();
1373
1374     this->tile_improvement_ptr = new Settlement(
1375         this->position_x,
1376         this->position_y,
1377         this->event_ptr,
1378         this->render_window_ptr,
1379         this->assets_manager_ptr,
1380         this->message_hub_ptr
1381     );
1382
1383     this->has_improvement = true;
1384
1385     this->assess();
1386     this->__sendAssessNeighboursMessage();
1387
1388     this->__sendUpdateGamePhaseMessage("system management");
1389     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1390     this->__sendTileStateMessage();
1391     this->__sendGameStateRequest();
1392
1393     return;
1394 } /* __buildSettlement() */

```

4.7.3.4 __buildSolarPV()

```

void HexTile::__buildSolarPV (
    void ) [private]

```

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

```

1452 {
1453     int build_cost = SOLAR_PV_BUILD_COST;
1454
1455     if (this->tile_type == TileType :: LAKE) {

```

```

1456         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1457     }
1458
1459     if (this->credits < build_cost) {
1460         std::cout << "Cannot build solar PV array: insufficient credits (need "
1461             << build_cost << " K)" << std::endl;
1462
1463         this->__sendInsufficientCreditsMessage();
1464         return;
1465     }
1466
1467     this->tile_improvement_ptr = new SolarPV(
1468         this->position_x,
1469         this->position_y,
1470         this->event_ptr,
1471         this->render_window_ptr,
1472         this->assets_manager_ptr,
1473         this->message_hub_ptr
1474     );
1475
1476     this->has_improvement = true;
1477     this->__closeBuildMenu();
1478
1479     if (this->tile_type == TileType::LAKE) {
1480         this->decoration_cleared = true;
1481         this->assets_manager_ptr->getSound("splash")->play();
1482     }
1483
1484     this->__sendCreditsSpentMessage(build_cost);
1485     this->__sendTileStateMessage();
1486     this->__sendGameStateRequest();
1487
1488     return;
1489 } /* __buildSolarPV() */

```

4.7.3.5 __buildTidalTurbine()

```

void HexTile::__buildTidalTurbine (
    void ) [private]

```

Helper method to build a tidal turbine on this tile.

```

1562 {
1563     int build_cost = TIDAL_TURBINE_BUILD_COST;
1564
1565     if (this->credits < build_cost) {
1566         std::cout << "Cannot build tidal turbine: insufficient credits (need "
1567             << build_cost << " K)" << std::endl;
1568
1569         this->__sendInsufficientCreditsMessage();
1570         return;
1571     }
1572
1573     this->tile_improvement_ptr = new TidalTurbine(
1574         this->position_x,
1575         this->position_y,
1576         this->event_ptr,
1577         this->render_window_ptr,
1578         this->assets_manager_ptr,
1579         this->message_hub_ptr
1580     );
1581
1582     this->has_improvement = true;
1583     this->decoration_cleared = true;
1584     this->assets_manager_ptr->getSound("splash")->play();
1585     this->__closeBuildMenu();
1586
1587     this->__sendCreditsSpentMessage(build_cost);
1588     this->__sendTileStateMessage();
1589     this->__sendGameStateRequest();
1590
1591     return;
1592 } /* __buildTidalTurbine() */

```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1607 {
1608     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1609
1610     if (this->credits < build_cost) {
1611         std::cout << "Cannot build wave energy converter: insufficient credits (need "
1612             << build_cost << " K)" << std::endl;
1613
1614         this->__sendInsufficientCreditsMessage();
1615         return;
1616     }
1617
1618     this->tile_improvement_ptr = new WaveEnergyConverter(
1619         this->position_x,
1620         this->position_y,
1621         this->event_ptr,
1622         this->render_window_ptr,
1623         this->assets_manager_ptr,
1624         this->message_hub_ptr
1625     );
1626
1627     this->has_improvement = true;
1628     this->decoration_cleared = true;
1629     this->assets_manager_ptr->getSound("splash")->play();
1630     this->__closeBuildMenu();
1631
1632     this->__sendCreditsSpentMessage(build_cost);
1633     this->__sendTileStateMessage();
1634     this->__sendGameStateRequest();
1635
1636     return;
1637 } /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1504 {
1505     int build_cost = WIND_TURBINE_BUILD_COST;
1506
1507     if (
1508         (this->tile_type == TileType :: LAKE) or
1509         (this->tile_type == TileType :: OCEAN)
1510     ) {
1511         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1512     }
1513
1514     if (this->credits < build_cost) {
1515         std::cout << "Cannot build wind turbine: insufficient credits (need "
1516             << build_cost << " K)" << std::endl;
1517
1518         this->__sendInsufficientCreditsMessage();
1519         return;
1520     }
1521
1522     this->tile_improvement_ptr = new WindTurbine(
1523         this->position_x,
1524         this->position_y,
1525         this->event_ptr,
1526         this->render_window_ptr,
1527         this->assets_manager_ptr,
1528         this->message_hub_ptr
1529     );
1530
1531     this->has_improvement = true;
1532     this->__closeBuildMenu();
1533
1534     if (
```



```

1535         (this->tile_type == TileType :: LAKE) or
1536         (this->tile_type == TileType :: OCEAN)
1537     ) {
1538         this->decoration_cleared = true;
1539         this->assets_manager_ptr->getSound("splash")->play();
1540     }
1541
1542     this->__sendCreditsSpentMessage(build_cost);
1543     this->__sendTileStateMessage();
1544     this->__sendGameStateRequest();
1545
1546     return;
1547 } /* __buildWindTurbine() */

```

4.7.3.8 __clearDecoration()

```

void HexTile::__clearDecoration (
    void ) [private]

```

Helper method to clear tile decoration.

```

791 {
792     this->decoration_cleared = true;
793     this->draw_explosion = true;
794
795     switch (this->tile_type) {
796         case (TileType :: FOREST): {
797             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
798
799             break;
800         }
801
802         case (TileType :: MOUNTAINS): {
803             this->assets_manager_ptr->getSound("clear mountains tile")->play();
804
805             break;
806         }
807
808         case (TileType :: PLAINS): {
809             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
810
811             break;
812         }
813
814         default: {
815             // do nothing!
816
817             break;
818         }
819     }
820
821     return;
822 } /* __clearDecoration() */

```

4.7.3.9 __closeBuildMenu()

```

void HexTile::__closeBuildMenu (
    void ) [private]

```

Helper method to close the tile improvement build menu.

```

1338 {
1339     if (not this->build_menu_open) {
1340         return;
1341     }
1342
1343     this->build_menu_open = false;
1344     this->assets_manager_ptr->getSound("build menu close")->play();
1345
1346     return;
1347 } /* __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.

```
1798 {
1799     std::string coords_substring = "TILE COORDS:  ";
1800     coords_substring += std::to_string(int(this->position_x - 400));
1801     coords_substring += ", ";
1802     coords_substring += std::to_string(int(this->position_y - 400));
1803     coords_substring += "\n";
1804
1805     return coords_substring;
1806 } /* __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.

```
1957 {
1958     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1959
1960     if (this->has_improvement) {
1961         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1962         improvement_substring += "\n";
1963     }
1964
1965     else {
1966         improvement_substring += "NONE\n";
1967     }
1968
1969     return improvement_substring;
1970 } /* __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.

```

1987 {
1988     //          32 char x 17 line console "-----\n";
1989     std::string options_substring          = "      **** TILE OPTIONS **** \n";
1990     options_substring                     += " \n";
1991
1992     if (this->game_phase == "build settlement") {
1993         if (
1994             (this->tile_type != TileType :: OCEAN) and
1995             (this->tile_type != TileType :: LAKE)
1996         ) {
1997             options_substring += "[B]:  BUILD SETTLEMENT (";
1998             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
1999             options_substring += " K)\n";
2000         }
2001     }
2002
2003
2004     else if (this->game_phase == "system management") {
2005         if (this->has_improvement) {
2006             options_substring.clear();
2007             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2008         }
2009
2010
2011         else if (not this->resource_assessed) {
2012             options_substring += "[A]:  ASSESS RESOURCE (";
2013             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2014             options_substring += " K)\n";
2015         }
2016
2017
2018         else if (
2019             (not this->decoration_cleared) and
2020             (this->tile_type != TileType :: OCEAN) and
2021             (this->tile_type != TileType :: LAKE)
2022         ) {
2023             options_substring += "[C]:  CLEAR TILE (";
2024
2025             switch (this->tile_type) {
2026                 case (TileType :: FOREST): {
2027                     options_substring += std::to_string(CLEAR_FOREST_COST);
2028
2029                     break;
2030                 }
2031
2032
2033                 case (TileType :: MOUNTAINS): {
2034                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2035
2036                     break;
2037                 }
2038
2039
2040                 case (TileType :: PLAINS): {
2041                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2042
2043                     break;
2044                 }
2045
2046
2047                 default: {
2048                     //do nothing!
2049
2050                     break;
2051                 }
2052             }
2053
2054             options_substring += " K)\n";
2055         }
2056
2057
2058         else if (
2059             (this->decoration_cleared) or
2060             (this->tile_type == TileType :: OCEAN) or
2061             (this->tile_type == TileType :: LAKE)
2062         ) {
2063             options_substring += "[B]:  OPEN BUILD MENU\n";
2064         }
2065     }
2066
2067
2068     else if (this->game_phase == "victory") {
2069         options_substring          += "      **** VICTORY **** \n";
2070     }

```

```

2071
2072
2073     else {
2074         options_substring += "        **** LOSS ****        \n";
2075     }
2076
2077     return options_substring;
2078 } /* __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.

```

1887 {
1888     std::string resource_substring = "TILE RESOURCE:        ";
1889
1890     if (this->resource_assessed) {
1891         switch (this->tile_resource) {
1892             case (TileResource :: POOR): {
1893                 resource_substring += "POOR\n";
1894
1895                 break;
1896             }
1897
1898             case (TileResource ::BELOW_AVERAGE): {
1899                 resource_substring += "BELOW AVERAGE\n";
1900
1901                 break;
1902             }
1903
1904             case (TileResource :: AVERAGE): {
1905                 resource_substring += "AVERAGE\n";
1906
1907                 break;
1908             }
1909
1910             case (TileResource :: ABOVE_AVERAGE): {
1911                 resource_substring += "ABOVE AVERAGE\n";
1912
1913                 break;
1914             }
1915
1916             case (TileResource :: GOOD): {
1917                 resource_substring += "GOOD\n";
1918
1919                 break;
1920             }
1921
1922             default: {
1923                 resource_substring += "???\n";
1924
1925                 break;
1926             }
1927         }
1928     }
1929
1930     else {
1931         resource_substring += "???\n";
1932     }
1933
1934     return resource_substring;
1935 } /* __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.

```
1823 {
1824     std::string type_substring = "TILE TYPE:      ";
1825
1826     switch (this->tile_type) {
1827         case (TileType :: FOREST): {
1828             type_substring += "FOREST\n";
1829
1830             break;
1831         }
1832
1833         case (TileType :: LAKE): {
1834             type_substring += "LAKE\n";
1835
1836             break;
1837         }
1838
1839         case (TileType :: MOUNTAINS): {
1840             type_substring += "MOUNTAINS\n";
1841
1842             break;
1843         }
1844
1845         case (TileType :: OCEAN): {
1846             type_substring += "OCEAN\n";
1847
1848             break;
1849         }
1850
1851         case (TileType :: PLAINS): {
1852             type_substring += "PLAINS\n";
1853
1854             break;
1855         }
1856
1857         default: {
1858             type_substring += "???\n";
1859
1860             break;
1861         }
1862     }
1863
1864     return type_substring;
1865 }
1866
1867 /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
874 {
875     if (not this->is_selected) {
876         return;
877     }
878
879     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
```

```
881         this->__setIsSelected(false);
882     }
883
884
885     if (this->build_menu_open) {
886         switch (this->tile_type) {
887             case (TileType :: FOREST): {
888                 switch (this->event_ptr->key.code) {
889                     case (sf::Keyboard::D): {
890                         this->__buildDieselGenerator();
891
892                         break;
893                     }
894
895
896                     case (sf::Keyboard::S): {
897                         this->__buildSolarPV();
898
899                         break;
900                     }
901
902
903                     case (sf::Keyboard::W): {
904                         this->__buildWindTurbine();
905
906                         break;
907                     }
908
909
910                     case (sf::Keyboard::E): {
911                         this->__buildEnergyStorage();
912
913                         break;
914                     }
915
916
917                     default: {
918                         // do nothing!
919
920                         break;
921                     }
922                 }
923
924                 break;
925             }
926
927
928             case (TileType :: LAKE): {
929                 switch (this->event_ptr->key.code) {
930                     case (sf::Keyboard::S): {
931                         this->__buildSolarPV();
932
933                         break;
934                     }
935
936
937                     case (sf::Keyboard::W): {
938                         this->__buildWindTurbine();
939
940                         break;
941                     }
942
943
944                     default: {
945                         // do nothing!
946
947                         break;
948                     }
949                 }
950
951                 break;
952             }
953
954
955             case (TileType :: MOUNTAINS): {
956                 switch (this->event_ptr->key.code) {
957                     case (sf::Keyboard::D): {
958                         this->__buildDieselGenerator();
959
960                         break;
961                     }
962
963
964                     case (sf::Keyboard::S): {
965                         this->__buildSolarPV();
966
967                         break;
```

```
968         }
969
970
971         case (sf::Keyboard::W): {
972             this->__buildWindTurbine();
973
974             break;
975         }
976
977
978         case (sf::Keyboard::E): {
979             this->__buildEnergyStorage();
980
981             break;
982         }
983
984
985         default: {
986             // do nothing!
987
988             break;
989         }
990     }
991
992     break;
993 }
994
995
996 case (TileType :: OCEAN): {
997     switch (this->event_ptr->key.code) {
998         case (sf::Keyboard::W): {
999             this->__buildWindTurbine();
1000
1001             break;
1002         }
1003
1004
1005         case (sf::Keyboard::T): {
1006             this->__buildTidalTurbine();
1007
1008             break;
1009         }
1010
1011
1012         case (sf::Keyboard::A): {
1013             this->__buildWaveEnergyConverter();
1014
1015             break;
1016         }
1017
1018
1019         default: {
1020             // do nothing!
1021
1022             break;
1023         }
1024     }
1025
1026     break;
1027 }
1028
1029
1030 case (TileType :: PLAINS): {
1031     switch (this->event_ptr->key.code) {
1032         case (sf::Keyboard::D): {
1033             this->__buildDieselGenerator();
1034
1035             break;
1036         }
1037
1038
1039         case (sf::Keyboard::S): {
1040             this->__buildSolarPV();
1041
1042             break;
1043         }
1044
1045
1046         case (sf::Keyboard::W): {
1047             this->__buildWindTurbine();
1048
1049             break;
1050         }
1051
1052
1053         case (sf::Keyboard::E): {
1054             this->__buildEnergyStorage();
```

```

1055
1056             break;
1057         }
1058
1059         default: {
1060             // do nothing!
1061
1062             break;
1063         }
1064     }
1065 }
1066
1067     break;
1068 }
1069
1070
1071     default: {
1072         //do nothing!
1073
1074         break;
1075     }
1076 }
1077 }
1078
1079
1080 if (this->game_phase == "build settlement") {
1081     if (
1082         (this->tile_type != TileType :: OCEAN) and
1083         (this->tile_type != TileType :: LAKE)
1084     ) {
1085         if (this->event_ptr->key.code == sf::Keyboard::B) {
1086             this->__buildSettlement();
1087         }
1088     }
1089 }
1090
1091
1092 else if (this->game_phase == "system management") {
1093     if (this->has_improvement) {
1094         if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
1095     {
1096         if (this->event_ptr->key.code == sf::Keyboard::P) {
1097             this->__scrapImprovement();
1098         }
1099     }
1100
1101     /*
1102     * All other inputs will be caught and handled by
1103     * this->tile_improvement_ptr->processEvent()
1104     */
1105 }
1106
1107 else if (not this->resource_assessed) {
1108     if (this->event_ptr->key.code == sf::Keyboard::A) {
1109         if (this->credits < RESOURCE_ASSESSMENT_COST) {
1110             std::cout << "Cannot assess resource: insufficient credits (need "
1111                 << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
1112
1113             this->__sendInsufficientCreditsMessage();
1114         }
1115
1116         else {
1117             this->assess();
1118             this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1119             this->__sendTileStateMessage();
1120             this->__sendGameStateRequest();
1121         }
1122     }
1123 }
1124
1125
1126 else if (
1127     (not this->decoration_cleared) and
1128     (this->tile_type != TileType :: OCEAN) and
1129     (this->tile_type != TileType :: LAKE)
1130 ) {
1131     if (this->event_ptr->key.code == sf::Keyboard::C) {
1132         int clear_cost = 0;
1133
1134         switch (this->tile_type) {
1135             case (TileType :: FOREST): {
1136                 clear_cost = CLEAR_FOREST_COST;
1137
1138                 break;
1139             }
1140

```



```

1141
1142         case (TileType :: MOUNTAINS): {
1143             clear_cost = CLEAR_MOUNTAINS_COST;
1144
1145             break;
1146         }
1147
1148
1149         case (TileType :: PLAINS): {
1150             clear_cost = CLEAR_PLAINS_COST;
1151
1152             break;
1153         }
1154
1155
1156         default: {
1157             // do nothing!
1158
1159             break;
1160         }
1161     }
1162
1163     if (this->credits < clear_cost) {
1164         std::cout << "Cannot clear tile: insufficient credits (need "
1165             << clear_cost << " K)" << std::endl;
1166
1167         this->__sendInsufficientCreditsMessage();
1168     }
1169
1170     else {
1171         this->__clearDecoration();
1172         this->__sendCreditsSpentMessage(clear_cost);
1173         this->__sendTileStateMessage();
1174         this->__sendGameStateRequest();
1175     }
1176 }
1177
1178
1179
1180     else if (
1181         (this->decoration_cleared) or
1182         (this->tile_type == TileType :: OCEAN) or
1183         (this->tile_type == TileType :: LAKE)
1184     ) {
1185         if (this->event_ptr->key.code == sf::Keyboard::B) {
1186             this->__openBuildMenu();
1187         }
1188     }
1189 }
1190
1191 return;
1192 } /* __handleKeyPressEvents() */

```

4.7.3.16 __handleKeyReleaseEvents()

```

void HexTile::__handleKeyReleaseEvents (
    void ) [private]
1198 {
1199     if (not this->is_selected) {
1200         return;
1201     }
1202
1203
1204     switch (this->event_ptr->key.code) {
1205         case (sf::Keyboard::P): {
1206             if (this->has_improvement) {
1207                 this->scrap_improvement_frame = 0;
1208
1209                 if (
1210                     this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1211                 ) {
1212                     this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1213                         sf::Color(255, 255, 255, 255)
1214                     );
1215                 }
1216
1217                 else {
1218                     for (
1219                         size_t i = 0;

```

```

1220             i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1221             i++;
1222         } {
1223             this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1224                 sf::Color(255, 255, 255)
1225             );
1226         }
1227     }
1228 }
1229
1230
1231     break;
1232 }
1233
1234
1235     default: {
1236         // do nothing!
1237
1238         break;
1239     }
1240 }
1241
1242 /*
1243 if (this->event_ptr->key.code == sf::Keyboard::P) {
1244
1245 }
1246 */
1247
1248 return;
1249 } /* __handleKeyReleaseEvents() */

```

4.7.3.17 __handleMouseButtonEvents()

```

void HexTile::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

1262 {
1263     switch (this->event_ptr->mouseButton.button) {
1264         case (sf::Mouse::Left): {
1265             if (this->__isClicked()) {
1266                 std::cout << "Tile (" << this->position_x << ", " <<
1267                     this->position_y << ") was selected" << std::endl;
1268
1269                 this->__setIsSelected(true);
1270
1271                 this->__sendTileSelectedMessage();
1272                 this->__sendTileStateMessage();
1273                 this->__sendGameStateRequest();
1274             }
1275
1276             else {
1277                 this->__setIsSelected(false);
1278             }
1279
1280             break;
1281         }
1282
1283         case (sf::Mouse::Right): {
1284             this->__setIsSelected(false);
1285
1286             break;
1287         }
1288
1289         default: {
1290             // do nothing!
1291
1292             break;
1293         }
1294     }
1295
1296     return;
1297 } /* __handleMouseButtonEvents() */

```

4.7.3.18 __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.

```
842 {
843     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
844
845     double mouse_x = mouse_position.x;
846     double mouse_y = mouse_position.y;
847
848     double distance = sqrt(
849         pow(this->position_x - mouse_x, 2) +
850         pow(this->position_y - mouse_y, 2)
851     );
852
853     if (distance < this->minor_radius) {
854         return true;
855     }
856     else {
857         return false;
858     }
859 } /* __isClicked() */
```

4.7.3.19 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

```
1314 {
1315     if (this->build_menu_open) {
1316         return;
1317     }
1318
1319     this->build_menu_open = true;
1320     this->assets_manager_ptr->getSound("build menu open")->play();
1321
1322     return;
1323 } /* __openBuildMenu() */
```

4.7.3.20 __scrapImprovement()

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

Helper method to scrap the tile improvement ([Settlement](#) cannot be scrapped). Requires the mapped key to be held continuously to confirm.

```
1697 {
1698     // 1. implement key hold confirmation
1699     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1700         double colour_scalar =
1701             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1702
1703         if (
1704             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1705         ) {
```

```

1706         this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1707             sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1708         );
1709     }
1710
1711     else {
1712         for (
1713             size_t i = 0;
1714             i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1715             i++
1716         ) {
1717             this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1718                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1719             );
1720         }
1721     }
1722
1723     this->scrap_improvement_frame += 4;
1724 }
1725
1726
1727 // 2. carry out scrapping
1728 else {
1729     this->draw_explosion = true;
1730     this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1731
1732     if (this->tile_improvement_ptr->production_menu_open) {
1733         this->tile_improvement_ptr->production_menu_open = false;
1734         this->assets_manager_ptr->getSound("build menu close")->play();
1735     }
1736
1737     delete this->tile_improvement_ptr;
1738     this->tile_improvement_ptr = NULL;
1739
1740     this->has_improvement = false;
1741
1742     this->scrap_improvement_frame = 0;
1743
1744     if (
1745         (this->tile_type == TileType :: LAKE) or
1746         (this->tile_type == TileType :: OCEAN)
1747     ) {
1748         this->decoration_cleared = false;
1749     }
1750
1751     this->__sendCreditsSpentMessage(SCRAP_COST);
1752     this->__sendTileStateMessage();
1753     this->__sendGameStateRequest();
1754 }
1755
1756 return;
1757 } /* __scrapImprovement() */

```

4.7.3.21 __sendAssessNeighboursMessage()

```

void HexTile::__sendAssessNeighboursMessage (
    void ) [private]

```

Helper method to format and send assess neighbours message.

```

2134 {
2135     Message assess_neighbours_message;
2136
2137     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2138     assess_neighbours_message.subject = "assess neighbours";
2139
2140     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2141
2142     std::cout << "Assess neighbours message sent by " << this << std::endl;
2143
2144     return;
2145 } /* __sendAssessNeighboursMessage() */

```

4.7.3.22 __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.
----------------------	--

```
2217 {
2218     Message credits_spent_message;
2219
2220     credits_spent_message.channel = GAME_CHANNEL;
2221     credits_spent_message.subject = "credits spent";
2222
2223     credits_spent_message.int_payload["credits spent"] = credits_spent;
2224
2225     this->message_hub_ptr->sendMessage(credits_spent_message);
2226
2227     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2228         << std::endl;
2229     return;
2230 } /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2160 {
2161     Message game_state_request;
2162
2163     game_state_request.channel = GAME_CHANNEL;
2164     game_state_request.subject = "state request";
2165
2166     this->message_hub_ptr->sendMessage(game_state_request);
2167
2168     std::cout << "Game state request message sent by " << this << std::endl;
2169     return;
2170 } /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2245 {
2246     Message insufficient_credits_message;
2247
2248     insufficient_credits_message.channel = GAME_CHANNEL;
2249     insufficient_credits_message.subject = "insufficient credits";
2250
2251     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2252
2253     std::cout << "Insufficient credits message sent by " << this << std::endl;
2254
2255     return;
2256 } /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1772 {
1773     Message tile_selected_message;
1774
1775     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1776     tile_selected_message.subject = "tile selected";
1777
1778     this->message_hub_ptr->sendMessage(tile_selected_message);
1779
1780     return;
1781 } /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2093 {
2094     Message tile_state_message;
2095
2096     tile_state_message.channel = TILE_STATE_CHANNEL;
2097     tile_state_message.subject = "tile state";
2098
2099
2100     //          32 char x 17 line console "-----\n";
2101     std::string console_string = "          **** TILE INFO **** \n";
2102
2103     console_string += this->__getTileCoordsSubstring();
2104     console_string += "          \n";
2105
2106     console_string += this->__getTileTypeSubstring();
2107     console_string += this->__getTileResourceSubstring();
2108     console_string += this->__getTileImprovementSubstring();
2109     console_string += "          \n";
2110
2111     console_string += this->__getTileOptionsSubstring();
2112
2113     tile_state_message.string_payload["console string"] = console_string;
2114
2115     this->message_hub_ptr->sendMessage(tile_state_message);
2116
2117     std::cout << "Tile state message sent by " << this << std::endl;
2118     return;
2119 } /* __sendTileStateMessage() */
```

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

```

2187 {
2188     Message update_game_phase_message;
2189
2190     update_game_phase_message.channel = GAME_CHANNEL;
2191     update_game_phase_message.subject = "update game phase";
2192
2193     update_game_phase_message.string_payload["game phase"] = game_phase;
2194
2195     this->message_hub_ptr->sendMessage(update_game_phase_message);
2196
2197     std::cout << "Update game phase message sent by " << this << std::endl;
2198
2199     return;
2200 } /* __sendUpdateGamePhaseMessage() */

```

4.7.3.28 __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.
--------------------	--

```

764 {
765     this->is_selected = is_selected;
766
767     if (this->tile_improvement_ptr != NULL) {
768         this->tile_improvement_ptr->setIsSelected(is_selected);
769     }
770
771     if ((not is_selected) and this->build_menu_open) {
772         this->__closeBuildMenu();
773     }
774
775     return;
776 } /* __setIsSelected() */

```

4.7.3.29 __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(MONOCROME_TEXT_GREEN);
223
224             break;
225         }
226
227         case (TileResource :: GOOD): {
228             this->resource_text.setString("+2");
229             this->resource_text.setFillColor(MONOCROME_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() */

```

4.7.3.30 __setUpBuildMenu()

```

void HexTile::__setUpBuildMenu (
    void ) [private]

```

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

```

667 {
668     this->build_menu_options_vec.clear();
669     this->build_menu_options_text_vec.clear();
670
671     // 1. set up and place build menu backing and text
672     this->build_menu_backing.setSize(sf::Vector2f(600, 256));
673     this->build_menu_backing.setOrigin(300, 128);
674     this->build_menu_backing.setPosition(400, 400);
675     this->build_menu_backing.setFillColor(MONOCROME_SCREEN_BACKGROUND);
676     this->build_menu_backing.setOutlineColor(MENU_FRAME_GREY);
677     this->build_menu_backing.setOutlineThickness(4);
678
679     this->build_menu_backing_text.setString("**** BUILD MENU ****");
680     this->build_menu_backing_text.setFont(
681         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
682     );
683     this->build_menu_backing_text.setCharacterSize(16);
684     this->build_menu_backing_text.setFillColor(MONOCROME_TEXT_GREEN);
685     this->build_menu_backing_text.setOrigin(
686         this->build_menu_backing_text.getLocalBounds().width / 2, 0
687     );

```



```

688     this->build_menu_backing_text.setPosition(400, 400 - 128 + 4);
689
690     // 2. set up and place build menu option sprites and text
691     switch (this->tile_type) {
692     case (TileType :: FOREST): {
693         this->__setUpDieselGeneratorBuildOption();
694         this->__setUpSolarPVBuildOption();
695         this->__setUpWindTurbineBuildOption();
696         //this->__setUpEnergyStorageSystemBuildOption();
697
698         break;
699     }
700
701
702     case (TileType :: LAKE): {
703         this->__setUpSolarPVBuildOption(true);
704         this->__setUpWindTurbineBuildOption(true);
705
706         break;
707     }
708
709
710     case (TileType :: MOUNTAINS): {
711         this->__setUpDieselGeneratorBuildOption();
712         this->__setUpSolarPVBuildOption();
713         this->__setUpWindTurbineBuildOption();
714         //this->__setUpEnergyStorageSystemBuildOption();
715
716         break;
717     }
718
719
720     case (TileType :: OCEAN): {
721         this->__setUpWindTurbineBuildOption(false, true);
722         this->__setUpTidalTurbineBuildOption();
723         this->__setUpWaveEnergyConverterBuildOption();
724
725         break;
726     }
727
728
729     case (TileType :: PLAINS): {
730         this->__setUpDieselGeneratorBuildOption();
731         this->__setUpSolarPVBuildOption();
732         this->__setUpWindTurbineBuildOption();
733         //this->__setUpEnergyStorageSystemBuildOption();
734
735         break;
736     }
737
738
739     default: {
740         // do nothing!
741
742         break;
743     }
744 }
745
746 return;
747 } /* __setUpBuildMenu() */

```

4.7.3.31 __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(MONOCHROME_TEXT_GREEN);
416
417     return;
418 } /* __setUpBuildOption() */

```

4.7.3.32 __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() */

```

4.7.3.33 __setUpEnergyStorageSystemBuildOption()

```

void HexTile::__setUpEnergyStorageSystemBuildOption (
    void ) [private]

```

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

```

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

```

4.7.3.34 __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() */

```

4.7.3.35 __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() */
```

4.7.3.36 __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() */
```

4.7.3.37 __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() */
```

4.7.3.38 __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     // ----- \n"
532     std::string solar_PV_string = " SOLAR PV ARRAY \n";
533     solar_PV_string += " \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() */
```

4.7.3.39 __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     // ----- \n"
573     std::string tidal_turbine_string = " TIDAL TURBINE \n";
574     tidal_turbine_string += " \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\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() */
```

4.7.3.40 __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() */
```

4.7.3.41 __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() */
```

4.7.3.42 __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     // -----
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() */

```

4.7.3.43 __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     // -----
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() */

```

4.7.3.44 assess()

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

Method to assess the tile's resource.

```
2679 {
2680     this->resource_assessed = true;
2681     this->resource_assessment = true;
2682
2683     this->assets_manager_ptr->getSound("resource assessment")->play();
2684
2685     this->__setResourceText();
2686     this->__sendTileStateMessage();
2687
2688     return;
2689 } /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2557 {
2558     switch (this->tile_type) {
2559         case (TileType :: FOREST): {
2560             this->tile_decoration_sprite.setTexture(
2561                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2562             );
2563
2564             break;
2565         }
2566
2567         case (TileType :: LAKE): {
2568             this->tile_decoration_sprite.setTexture(
2569                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2570             );
2571
2572             break;
2573         }
2574
2575         case (TileType :: MOUNTAINS): {
2576             this->tile_decoration_sprite.setTexture(
2577                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2578             );
2579
2580             break;
2581         }
2582
2583         case (TileType :: OCEAN): {
2584             this->tile_decoration_sprite.setTexture(
2585                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2586             );
2587
2588             break;
2589         }
2590
2591         case (TileType :: PLAINS): {
2592             this->tile_decoration_sprite.setTexture(
2593                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2594             );
2595
2596             break;
2597         }
2598
2599         default: {
2600             // do nothing!
2601
2602             break;
2603         }
2604     }
2605
2606     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```



```

2608         this->tile_decoration_sprite.setOrigin(
2609             this->tile_decoration_sprite.getLocalBounds().width / 2,
2610             this->tile_decoration_sprite.getLocalBounds().height / 2
2611         );
2612
2613         this->tile_decoration_sprite.setPosition(
2614             this->position_x,
2615             this->position_y
2616         );
2617
2618         if ((double)rand() / RAND_MAX > 0.5) {
2619             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2620         }
2621     }
2622
2623     else {
2624         this->tile_decoration_sprite.setOrigin(
2625             this->tile_decoration_sprite.getLocalBounds().width / 2,
2626             this->tile_decoration_sprite.getLocalBounds().height
2627         );
2628
2629         this->tile_decoration_sprite.setPosition(
2630             this->position_x,
2631             this->position_y + 12
2632         );
2633
2634         if ((double)rand() / RAND_MAX > 0.5) {
2635             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2636         }
2637     }
2638
2639     return;
2640 } /* decorateTile(void) */

```

4.7.3.46 draw()

```

void HexTile::draw (
    void )

```

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

```

2807 {
2808     // 1. draw hex
2809     this->render_window_ptr->draw(this->tile_sprite);
2810
2811     // 2. draw node
2812     if (this->show_node) {
2813         this->render_window_ptr->draw(this->node_sprite);
2814     }
2815
2816     // 3. draw tile decoration
2817     if (not this->decoration_cleared) {
2818         this->render_window_ptr->draw(this->tile_decoration_sprite);
2819     }
2820
2821     // 4. draw selection outline
2822     if (this->is_selected) {
2823         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2824
2825         outline_colour.a =
2826             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2827
2828         this->select_outline_sprite.setOutlineColor(outline_colour);
2829
2830         this->render_window_ptr->draw(this->select_outline_sprite);
2831     }
2832
2833     // 5. draw tile improvement
2834     if (this->has_improvement) {
2835         if (not this->tile_improvement_ptr->just_built) {
2836             this->tile_improvement_ptr->draw();
2837         }
2838     }
2839
2840     // 6. draw resource
2841     if (this->show_resource) {
2842         this->render_window_ptr->draw(this->resource_chip_sprite);
2843         this->render_window_ptr->draw(this->resource_text);
2844     }

```

```

2845
2846 // 7. draw resource assessment notification
2847 if (this->resource_assessment) {
2848     int alpha = this->magnifying_glass_sprite.getColor().a;
2849
2850     alpha -= 0.05 * FRAMES_PER_SECOND;
2851     if (alpha < 0) {
2852         alpha = 0;
2853         this->resource_assessment = false;
2854     }
2855
2856     this->magnifying_glass_sprite.setColor(
2857         sf::Color(255, 255, 255, alpha)
2858     );
2859
2860     this->render_window_ptr->draw(this->magnifying_glass_sprite);
2861 }
2862
2863 // 8. draw explosion, then settlement placement
2864 if (this->draw_explosion) {
2865     this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2866
2867     if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2868         this->explosion_frame++;
2869     }
2870
2871     if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2872         this->draw_explosion = false;
2873         this->explosion_frame = 0;
2874     }
2875 }
2876
2877 else if (this->has_improvement) {
2878     if (this->tile_improvement_ptr->just_built) {
2879         this->tile_improvement_ptr->draw();
2880     }
2881 }
2882
2883 // 9. build menu
2884 if (this->build_menu_open) {
2885     this->render_window_ptr->draw(this->build_menu_backing);
2886     this->render_window_ptr->draw(this->build_menu_backing_text);
2887
2888     for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2889         for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2890             this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2891         }
2892         this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2893     }
2894 }
2895
2896 this->frame++;
2897 return;
2898 } /* draw() */

```

4.7.3.47 processEvent()

```

void HexTile::processEvent (
    void )

```

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

```

2704 {
2705     // 1. process TileImprovement events
2706     if (
2707         this->is_selected and
2708         this->tile_improvement_ptr != NULL
2709     ) {
2710         this->tile_improvement_ptr->processEvent();
2711     }
2712
2713     // 2. process HexTile events
2714     if (this->event_ptr->type == sf::Event::KeyPressed) {
2715         this->__handleKeyPressEvents();
2716     }
2717
2718     if (this->event_ptr->type == sf::Event::KeyReleased) {
2719         this->__handleKeyReleaseEvents();
2720     }

```

```

2721
2722     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2723         this->__handleMouseButtonEvents();
2724     }
2725
2726     return;
2727 } /* processEvent() */

```

4.7.3.48 processMessage()

```

void HexTile::processMessage (
    void )

```

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

```

2742 {
2743     // 1. process TileImprovement messages
2744     if (
2745         this->is_selected and
2746         this->tile_improvement_ptr != NULL
2747     ) {
2748         this->tile_improvement_ptr->processMessage();
2749     }
2750
2751     // 2. process HexTile messages
2752     if (this->is_selected) {
2753         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2754             Message game_state_message = this->message_hub_ptr->receiveMessage(
2755                 GAME_STATE_CHANNEL
2756             );
2757
2758             if (game_state_message.subject == "game state") {
2759                 this->credits = game_state_message.int_payload["credits"];
2760                 this->game_phase = game_state_message.string_payload["game phase"];
2761
2762                 if (this->tile_improvement_ptr != NULL) {
2763                     this->tile_improvement_ptr->credits = this->credits;
2764                     this->tile_improvement_ptr->game_phase = this->game_phase;
2765                 }
2766
2767                 std::cout << "Game state message received by " << this << std::endl;
2768                 this->__sendTileStateMessage();
2769                 this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
2770             }
2771         }
2772
2773         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2774             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2775                 TILE_STATE_CHANNEL
2776             );
2777
2778             if (tile_state_message.subject == "state request") {
2779                 this->__sendTileStateMessage();
2780
2781                 std::cout << "Tile state request received by " << this << std::endl;
2782                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2783             }
2784         }
2785
2786         std::cout << "Current credits (HexTile): " << this->credits << " K" <<
2787             std::endl;
2788     }
2789
2790     return;
2791 } /* processMessage() */

```

4.7.3.49 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].
--------------------	--

```

2506 {
2507     // 1. check input
2508     if (input_value < 0 or input_value > 1) {
2509         std::string error_str = "ERROR HexTile::setTileResource() given input value is ";
2510         error_str += "not in the closed interval [0, 1]";
2511
2512         #ifdef _WIN32
2513             std::cout << error_str << std::endl;
2514         #endif /* _WIN32 */
2515
2516         throw std::runtime_error(error_str);
2517     }
2518
2519     // 2. convert input value to tile resource
2520     TileResource tile_resource;
2521
2522     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2523         tile_resource = TileResource :: POOR;
2524     }
2525     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2526         tile_resource = TileResource :: BELOW_AVERAGE;
2527     }
2528     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2529         tile_resource = TileResource :: AVERAGE;
2530     }
2531     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2532         tile_resource = TileResource :: ABOVE_AVERAGE;
2533     }
2534     else {
2535         tile_resource = TileResource :: GOOD;
2536     }
2537
2538     // 3. call alternate method
2539     this->setTileResource(tile_resource);
2540
2541     return;
2542 } /* setTileResource(double) */

```

4.7.3.50 setTileResource() [2/2]

```

void HexTile::setTileResource (
    TileResource tile_resource )

```

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

Parameters

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

```

2484 {
2485     this->tile_resource = tile_resource;
2486     this->__setResourceText();
2487
2488     return;
2489 } /* setTileResource(TileResource) */

```

4.7.3.51 setTileType() [1/2]

```

void HexTile::setTileType (
    double input_value )

```

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

Parameters

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

```

2434 {
2435     // 1. check input
2436     if (input_value < 0 or input_value > 1) {
2437         std::string error_str = "ERROR HexTile::setTileType() given input value is ";
2438         error_str += "not in the closed interval [0, 1]";
2439
2440         #ifdef _WIN32
2441             std::cout << error_str << std::endl;
2442         #endif /* _WIN32 */
2443
2444         throw std::runtime_error(error_str);
2445     }
2446
2447     // 2. convert input value to tile type
2448     TileType tile_type;
2449
2450     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2451         tile_type = TileType :: LAKE;
2452     }
2453     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2454         tile_type = TileType :: PLAINS;
2455     }
2456     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2457         tile_type = TileType :: FOREST;
2458     }
2459     else {
2460         tile_type = TileType :: MOUNTAINS;
2461     }
2462
2463     // 3. call alternate method
2464     this->setTileType(tile_type);
2465
2466     return;
2467 } /* setTileType(double) */

```

4.7.3.52 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.
------------------	---

```

2373 {
2374     this->tile_type = tile_type;
2375
2376     switch (this->tile_type) {
2377         case (TileType :: FOREST): {
2378             this->tile_sprite.setFillColor(FOREST_GREEN);
2379
2380             break;
2381         }
2382
2383         case (TileType :: LAKE): {
2384             this->tile_sprite.setFillColor(LAKE_BLUE);
2385
2386             break;
2387         }
2388
2389         case (TileType :: MOUNTAINS): {
2390             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2391
2392             break;
2393         }
2394
2395         case (TileType :: OCEAN): {

```

```

2396         this->tile_sprite.setFillColor(OCEAN_BLUE);
2397
2398         break;
2399     }
2400
2401     case (TileType :: PLAINS): {
2402         this->tile_sprite.setFillColor(PLAINS_YELLOW);
2403
2404         break;
2405     }
2406
2407     default: {
2408         // do nothing!
2409
2410         break;
2411     }
2412 }
2413
2414 this->__setUpBuildMenu();
2415
2416 return;
2417 } /* setTileType(TileType) */

```

4.7.3.53 toggleResourceOverlay()

```

void HexTile::toggleResourceOverlay (
    void )

```

Method to toggle the tile resource overlay.

```

2655 {
2656     if (this->show_resource) {
2657         this->show_resource = false;
2658     }
2659     else {
2660         this->show_resource = true;
2661     }
2662
2663     return;
2664 } /* 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 scrap_improvement_frame

```
int HexTile::scrap_improvement_frame
```

A frame for key-hold to confirm scrapping.

4.7.4.30 select_outline_sprite

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

A convex shape which outlines the tile when selected.

4.7.4.31 show_node

```
bool HexTile::show_node
```

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

4.7.4.32 show_resource

```
bool HexTile::show_resource
```

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

4.7.4.33 tile_decoration_sprite

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

A tile decoration sprite.

4.7.4.34 tile_improvement_ptr

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

A pointer to the improvement for this tile.

4.7.4.35 tile_resource

```
TileResource HexTile::tile_resource
```

4.7.4.36 tile_sprite

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

A convex shape which represents the tile.

4.7.4.37 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     return false;
111 } /* 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
177         throw std::runtime_error(error_str);
178     }
179
180     // 2. remove channel from map
181     this->message_map[channel].clear();
182     this->message_map.erase(channel);
183
184     std::cout << "Channel " << channel << " removed from message hub" << std::endl;
185
186     return;
187 } /* 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
218         throw std::runtime_error(error_str);
219     }
220
221     // 2. send message to message map
222     this->message_map[channel].push_back(message);
223
224     return;
225 } /* sendMessage() */

```

4.9.4 Member Data Documentation

4.9.4.1 message_map

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

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

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

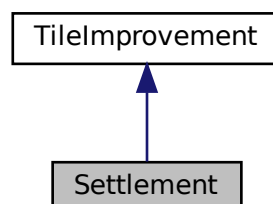
- header/ESC_core/[MessageHub.h](#)
- source/ESC_core/[MessageHub.cpp](#)

4.10 Settlement Class Reference

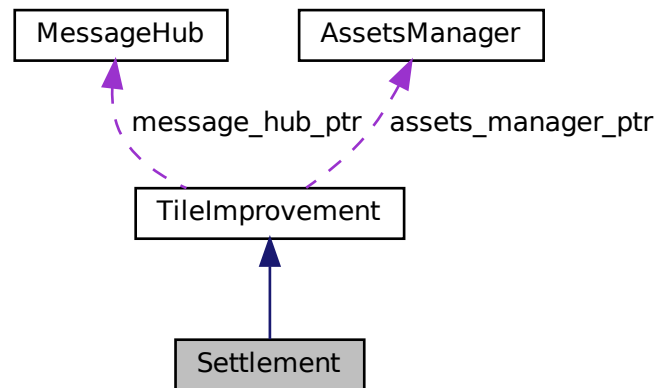
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for Settlement:



Collaboration diagram for Settlement:



Public Member Functions

- [Settlement](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [Settlement](#) class.
- void [setIsSelected](#) (bool)
Method to set the is selected attribute.
- 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.

```
209 :
210 TileImprovement (
211     position_x,
212     position_y,
213     event_ptr,
214     render_window_ptr,
215     assets_manager_ptr,
216     message_hub_ptr
217 )
```

```

218 {
219     // 1. set attributes
220
221     // 1.1. private
222     //...
223
224     // 1.2. public
225     this->tile_improvement_type = TileImprovementType :: SETTLEMENT;
226
227     this->smoke_da = SECONDS_PER_FRAME / 4;
228     this->smoke_dx = 5 * SECONDS_PER_FRAME;
229     this->smoke_dy = -10 * SECONDS_PER_FRAME;
230     this->smoke_prob = 3 * SECONDS_PER_FRAME;
231
232     this->smoke_sprite_list = {};
233
234     this->tile_improvement_string = "SETTLEMENT";
235
236     this->__setUpTileImprovementSpriteStatic();
237
238     std::cout << "Settlement constructed at " << this << std::endl;
239
240     return;
241 } /* Settlement() */

```

4.10.2.2 ~Settlement()

```

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

```

Destructor for the [Settlement](#) class.

```

440 {
441     std::cout << "Settlement at " << this << " destroyed" << std::endl;
442
443     return;
444 } /* ~Settlement() */

```

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

```

void Settlement::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

103 {
104     if (this->just_built) {
105         return;
106     }
107
108     switch (this->event_ptr->key.code) {
109         //...
110
111         default: {
112             // do nothing!
113
114             break;
115         }
116     }
117
118     return;
119 } /* __handleKeyPressEvents() */

```

4.10.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

```
359 {
360     // 1. if just built, call base method and return
361     if (this->just_built) {
362         TileImprovement :: draw();
363     }
364     return;
365 }
366
367 // 2. draw static sprite and chimney smoke effects
368 this->render_window_ptr->draw(this->tile_improvement_sprite_static);
369
370 std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
371
372 double alpha = 255;
373
374 while (iter != this->smoke_sprite_list.end()) {
375     this->render_window_ptr->draw(*iter);
376
377     alpha = (*iter).getColor().a;
378     alpha -= this->smoke_da;
379
380     if (alpha <= 0) {
381         iter = this->smoke_sprite_list.erase(iter);
382         continue;
383     }
384
385     (*iter).setColor(sf::Color(255, 255, 255, alpha));
386
387     (*iter).move(
388         this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
389         this->smoke_dy
390     );
391
392     (*iter).rotate((((double)rand() / RAND_MAX)));
393
394     iter++;
395 }
396
397
398
399 if ((double)rand() / RAND_MAX < smoke_prob) {
400     this->smoke_sprite_list.push_back(
401         sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
402     );
403
404     this->smoke_sprite_list.back().setOrigin(
405         this->smoke_sprite_list.back().getLocalBounds().width / 2,
406         this->smoke_sprite_list.back().getLocalBounds().height / 2
407     );
408
409     this->smoke_sprite_list.back().setPosition(
410         this->position_x + 9 + 4 * (((double)rand() / RAND_MAX) - 2),
411         this->position_y - 33
412     );
413 }
414
415 // 3. draw production menu
416 if (this->production_menu_open) {
417     this->render_window_ptr->draw(this->production_menu_backing);
418     this->render_window_ptr->draw(this->production_menu_backing_text);
419
420     //...
421 }
422
423 this->frame++;
424 return;
425 } /* 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](#).

```
283 {
284     //          32 char x 17 line console "-----\n";
285     std::string options_substring          = "    *** SETTLEMENT OPTIONS *** \n";
286     options_substring                     += " \n";
287     options_substring                     += " \n";
288     options_substring                     += " \n";
289     options_substring                     += " \n";
290     options_substring                     += " \n";
291     options_substring                     += " \n";
292     options_substring                     += " \n";
293
294     return options_substring;
295 } /* getTileOptionsSubstring() */
```

4.10.3.6 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
310 {
311     TileImprovement :: processEvent();
312
313     if (this->event_ptr->type == sf::Event::KeyPressed) {
314         this->__handleKeyPressEvents();
315     }
316
317     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
318         this->__handleMouseButtonEvents();
319     }
320
321     return;
322 } /* processEvent() */
```

4.10.3.7 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
337 {
338     TileImprovement :: processMessage();
339
340     //...
341
342     return;
343 } /* processMessage() */
```

4.10.3.8 setIsSelected()

```
void Settlement::setIsSelected (
    bool is_selected ) [virtual]
```

Method to set the is selected attribute.

Parameters

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

Reimplemented from [TileImprovement](#).

```
258 {
259     TileImprovement :: setIsSelected(is_selected);
260
261     if (this->is_selected) {
262         this->assets_manager_ptr->getSound("people and children")->play();
263     }
264
265     return;
266 } /* setIsSelected() */
```

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:

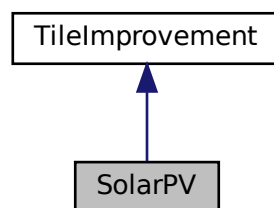
- header/[Settlement.h](#)
- source/[Settlement.cpp](#)

4.11 SolarPV Class Reference

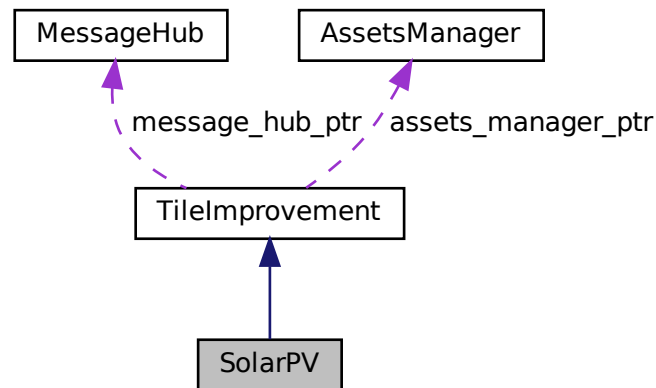
A settlement class (child class of [TileImprovement](#)).

```
#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.

Public Attributes

- int [capacity_kW](#)
The rated production capacity [kW] of the solar PV array.
- int [production_MWh](#)
The current production [MWh] of the solar PV array.
- int [dispatchable_MWh](#)
The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__upgradePowerCapacity](#) (void)
Helper method to upgrade power capacity.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__drawUpgradeOptions](#) (void)
Helper method to set up and draw upgrade options.

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.

```
396 :
397 TileImprovement (
```

```

398     position_x,
399     position_y,
400     event_ptr,
401     render_window_ptr,
402     assets_manager_ptr,
403     message_hub_ptr
404 )
405 {
406     // 1. set attributes
407
408     // 1.1. private
409     //...
410
411     // 1.2. public
412     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
413
414     this->is_running = false;
415
416     this->health = 100;
417
418     this->capacity_kW = 100;
419     this->upgrade_level = 1;
420     this->storage_level = 0;
421
422     this->production_MWh = 0;
423     this->dispatchable_MWh = 0;
424
425     this->tile_improvement_string = "SOLAR PV ARRAY";
426
427     this->__setUpTileImprovementSpriteStatic();
428
429     std::cout << "SolarPV constructed at " << this << std::endl;
430
431     return;
432 } /* SolarPV() */

```

4.11.2.2 ~SolarPV()

```

SolarPV::~SolarPV (
    void ) [virtual]

```

Destructor for the [SolarPV](#) class.

```

628 {
629     std::cout << "SolarPV at " << this << " destroyed" << std::endl;
630
631     return;
632 } /* ~SolarPV() */

```

4.11.3 Member Function Documentation

4.11.3.1 __drawUpgradeOptions()

```

void SolarPV::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

253 {
254     // 1. draw power capacity upgrade sprite
255     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
256     this->tile_improvement_sprite_static.setPosition(400 - 100, 400 - 32);
257
258     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
259     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
260
261     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();

```

```

262     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
263
264     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
265
266     this->tile_improvement_sprite_static.setPosition(initial_position);
267     this->tile_improvement_sprite_static.setColor(initial_colour);
268     this->tile_improvement_sprite_static.setScale(initial_scale);
269
270     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
271
272
273     // 2. draw power capacity upgrade text
274     //      16 char line = "          \n"
275     std::string power_upgrade_string = "POWER CAPACITY \n";
276     power_upgrade_string           += "          \n";
277
278     power_upgrade_string           += "CAPACITY: ";
279     power_upgrade_string           += std::to_string(this->capacity_kW);
280     power_upgrade_string           += " kW\n";
281
282     power_upgrade_string           += "LEVEL: ";
283     power_upgrade_string           += std::to_string(this->upgrade_level);
284     power_upgrade_string           += "\n\n";
285
286     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
287         power_upgrade_string       += "[W]: + 100 kW (";
288         power_upgrade_string       += std::to_string(SOLAR_PV_BUILD_COST);
289         power_upgrade_string       += " K)\n";
290     }
291
292     else {
293         power_upgrade_string       += " * MAX LEVEL * \n";
294     }
295
296     sf::Text power_upgrade_text = sf::Text(
297         power_upgrade_string,
298         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
299         16
300     );
301
302     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
303     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
304     power_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
305
306     this->render_window_ptr->draw(power_upgrade_text);
307
308
309     // 3. draw energy capacity (storage) upgrade sprite
310     this->render_window_ptr->draw(this->storage_upgrade_sprite);
311     this->render_window_ptr->draw(this->upgrade_plus_sprite);
312
313
314     // 4. draw energy capacity (storage) upgrade text
315     //      16 char line = "          \n"
316     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
317     energy_upgrade_string           += "          \n";
318
319     energy_upgrade_string           += "CAPACITY: ";
320     energy_upgrade_string           += std::to_string(this->storage_level * 200);
321     energy_upgrade_string           += " kWh\n";
322
323     energy_upgrade_string           += "LEVEL: ";
324     energy_upgrade_string           += std::to_string(this->storage_level);
325     energy_upgrade_string           += "\n\n";
326
327     if (this->storage_level < MAX_STORAGE_LEVELS) {
328         energy_upgrade_string       += "[D]: + 200 kWh (";
329         energy_upgrade_string       += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
330         energy_upgrade_string       += " K)\n";
331     }
332
333     else {
334         energy_upgrade_string += " * MAX LEVEL * \n";
335     }
336
337     sf::Text energy_upgrade_text = sf::Text(
338         energy_upgrade_string,
339         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
340         16
341     );
342
343     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
344     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
345     energy_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
346
347     this->render_window_ptr->draw(energy_upgrade_text);
348

```



```

349     return;
350 } /* __drawUpgradeOptions() */

```

4.11.3.2 __handleKeyPressEvents()

```

void SolarPV::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

145 {
146     if (this->just_built) {
147         return;
148     }
149
150     switch (this->event_ptr->key.code) {
151         case (sf::Keyboard::U): {
152             this->__openUpgradeMenu();
153
154             break;
155         }
156
157         case (sf::Keyboard::W): {
158             if (this->production_menu_open) {
159                 //...
160             }
161
162             else if (this->upgrade_menu_open) {
163                 this->__upgradePowerCapacity();
164             }
165
166             break;
167         }
168
169         case (sf::Keyboard::S): {
170             //...
171
172             break;
173         }
174
175         case (sf::Keyboard::D): {
176             if (this->upgrade_menu_open) {
177                 this->__upgradeStorageCapacity();
178             }
179
180             break;
181         }
182
183         default: {
184             // do nothing!
185
186             break;
187         }
188     }
189
190     return;
191 } /* __handleKeyPressEvents() */

```

4.11.3.3 __handleMouseButtonEvents()

```

void SolarPV::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

210 {
211     if (this->just_built) {

```

```

212         return;
213     }
214
215     switch (this->event_ptr->mouseButton.button) {
216     case (sf::Mouse::Left): {
217         //...
218
219         break;
220     }
221
222
223     case (sf::Mouse::Right): {
224         //...
225
226         break;
227     }
228
229
230     default: {
231         // do nothing!
232
233         break;
234     }
235     }
236
237     return;
238 } /* __handleMouseButtonEvents() */

```

4.11.3.4 __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
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() */

```

4.11.3.5 __upgradePowerCapacity()

```

void SolarPV::__upgradePowerCapacity (
    void ) [private]

```

Helper method to upgrade power capacity.

```

103 {
104     if (this->credits < SOLAR_PV_BUILD_COST) {
105         std::cout << "Cannot upgrade solar PV: insufficient credits (need "
106             << SOLAR_PV_BUILD_COST << " K)" << std::endl;
107
108         this->__sendInsufficientCreditsMessage();
109         return;

```

```

110     }
111
112     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
113         return;
114     }
115
116     this->health = 100;
117
118     this->capacity_kW += 100;
119     this->upgrade_level++;
120
121     this->just_upgraded = true;
122
123     this->assets_manager_ptr->getSound("upgrade")->play();
124
125     this->__sendCreditsSpentMessage(SOLAR_PV_BUILD_COST);
126     this->__sendTileStateRequest();
127     this->__sendGameStateRequest();
128
129     return;
130 } /* __upgradePowerCapacity() */

```

4.11.3.6 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](#).

```

543 {
544     // 1. if just built, call base method and return
545     if (this->just_built) {
546         TileImprovement :: draw();
547
548         return;
549     }
550
551
552     // 2. handle upgrade effects
553     if (this->just_upgraded) {
554         this->tile_improvement_sprite_static.setColor(
555             sf::Color(
556                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
557                 255,
558                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
559                 255
560             )
561         );
562
563         this->tile_improvement_sprite_static.setScale(
564             sf::Vector2f(
565                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
566                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
567             )
568         );
569
570         this->upgrade_frame++;
571     }
572
573     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
574         this->tile_improvement_sprite_static.setColor(
575             sf::Color(255,255,255,255)
576         );
577
578         this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
579
580         this->just_upgraded = false;
581         this->upgrade_frame = 0;
582     }
583
584
585     // 3. draw static sprite
586     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
587
588
589     // 4. draw storage upgrades

```

```

590     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
591         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
592     }
593
594
595     // 5. draw production menu
596     if (this->production_menu_open) {
597         this->render_window_ptr->draw(this->production_menu_backing);
598         this->render_window_ptr->draw(this->production_menu_backing_text);
599
600         //...
601     }
602
603     // 6. draw upgrade menu
604     if (this->upgrade_menu_open) {
605         this->render_window_ptr->draw(this->upgrade_menu_backing);
606         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
607
608         this->__drawUpgradeOptions();
609     }
610
611     this->frame++;
612     return;
613 } /* draw() */

```

4.11.3.7 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

449 {
450     //          32 char x 17 line console "-----\n";
451     std::string options_substring = "CAPACITY: ";
452     options_substring += std::to_string(this->capacity_kW);
453     options_substring += " kW (level ";
454     options_substring += std::to_string(this->upgrade_level);
455     options_substring += ")\n";
456
457     options_substring += "PRODUCTION: ";
458     options_substring += std::to_string(this->production_MWh);
459     options_substring += " MWh\n";
460
461     options_substring += "DISPATCHABLE: ";
462     options_substring += std::to_string(this->dispatchable_MWh);
463     options_substring += " MWh\n";
464
465     options_substring += "HEALTH: ";
466     options_substring += std::to_string(this->health);
467     options_substring += "/100\n";
468
469     options_substring += "
470     options_substring += "    **** SOLAR PV OPTIONS ****
471     options_substring += "
472     options_substring += "        [E]:  OPEN PRODUCTION MENU
473     options_substring += "        [U]:  OPEN UPGRADE MENU
474     options_substring += "HOLD [P]:  SCRAP (";
475     options_substring += std::to_string(SCRAP_COST);
476     options_substring += " K)";
477
478     return options_substring;
479 } /* getTileOptionsSubstring() */

```

4.11.3.8 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
494 {
495     TileImprovement :: processEvent();
496
497     if (this->event_ptr->type == sf::Event::KeyPressed) {
498         this->__handleKeyPressEvents();
499     }
500
501     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
502         this->__handleMouseButtonEvents();
503     }
504
505     return;
506 } /* processEvent() */
```

4.11.3.9 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
521 {
522     TileImprovement :: processMessage();
523
524     //...
525
526     return;
527 } /* processMessage() */
```

4.11.4 Member Data Documentation

4.11.4.1 capacity_kW

```
int SolarPV::capacity_kW
```

The rated production capacity [kW] of the solar PV array.

4.11.4.2 dispatchable_MWh

```
int SolarPV::dispatchable_MWh
```

The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

4.11.4.3 production_MWh

```
int SolarPV::production_MWh
```

The current production [MWh] of the solar PV array.

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

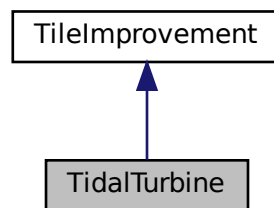
- header/[SolarPV.h](#)
- source/[SolarPV.cpp](#)

4.12 TidalTurbine Class Reference

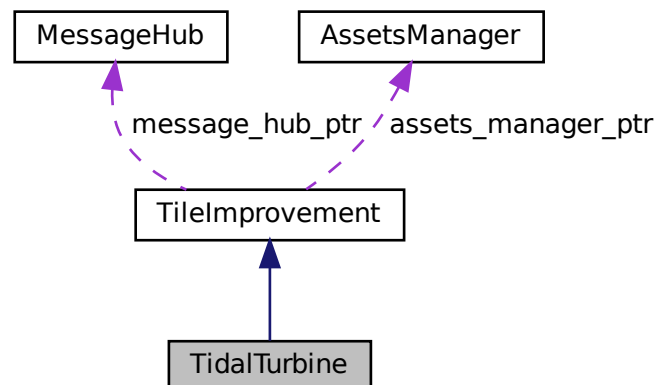
A settlement class (child class of [TileImprovement](#)).

```
#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.

Public Attributes

- int [capacity_kW](#)
The rated production capacity [kW] of the solar PV array.
- int [production_MWh](#)
The current production [MWh] of the solar PV array.
- int [dispatchable_MWh](#)
The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__upgradePowerCapacity](#) (void)
Helper method to upgrade power capacity.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__drawUpgradeOptions](#) (void)
Helper method to set up and draw upgrade options.

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.

```
409 :
410 TileImprovement (
411     position_x,
412     position_y,
413     event_ptr,
414     render_window_ptr,
415     assets_manager_ptr,
416     message_hub_ptr
417 )
418 {
419     // 1. set attributes
420
421     // 1.1. private
422     //...
423
424     // 1.2. public
425     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
426
427     this->is_running = false;
428
429     this->health = 100;
430
431     this->capacity_kW = 100;
432     this->upgrade_level = 1;
433     this->storage_level = 0;
434
435     this->production_MWh = 0;
436     this->dispatchable_MWh = 0;
437
438     this->tile_improvement_string = "TIDAL TURBINE";
439
440     this->__setUpTileImprovementSpriteAnimated();
441
442     std::cout << "TidalTurbine constructed at " << this << std::endl;
443
444     return;
445 } /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

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


Destructor for the [TidalTurbine](#) class.

```

658 {
659     std::cout << "TidalTurbine at " << this << " destroyed" << std::endl;
660
661     return;
662 } /* ~TidalTurbine() */

```

4.12.3 Member Function Documentation

4.12.3.1 __drawUpgradeOptions()

```

void TidalTurbine::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

264 {
265     // 1. draw power capacity upgrade sprite
266     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
267         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
268         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
269
270         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
271         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
272
273         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
274         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
275
276         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
277
278         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
279         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
280         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
281     }
282
283     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
284
285
286     // 2. draw power capacity upgrade text
287     //      16 char line = "
288     std::string power_upgrade_string = "POWER CAPACITY \n";
289     power_upgrade_string += "
290
291     power_upgrade_string += "CAPACITY: ";
292     power_upgrade_string += std::to_string(this->capacity_kW);
293     power_upgrade_string += " kW\n";
294
295     power_upgrade_string += "LEVEL: ";
296     power_upgrade_string += std::to_string(this->upgrade_level);
297     power_upgrade_string += "\n\n";
298
299     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
300         power_upgrade_string += "[W]: + 100 kW (";
301         power_upgrade_string += std::to_string(TIDAL_TURBINE_BUILD_COST);
302         power_upgrade_string += " K)\n";
303     }
304
305     else {
306         power_upgrade_string += " * MAX LEVEL * \n";
307     }
308
309     sf::Text power_upgrade_text = sf::Text(
310         power_upgrade_string,
311         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
312         16
313     );
314
315     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
316     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
317     power_upgrade_text.setFill_color(MONOCROME_TEXT_GREEN);
318
319     this->render_window_ptr->draw(power_upgrade_text);
320
321
322     // 3. draw energy capacity (storage) upgrade sprite

```

```

323     this->render_window_ptr->draw(this->storage_upgrade_sprite);
324     this->render_window_ptr->draw(this->upgrade_plus_sprite);
325
326
327     // 4. draw energy capacity (storage) upgrade text
328     //          16 char line = "          \n"
329     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
330     energy_upgrade_string            += "          \n";
331
332     energy_upgrade_string            += "CAPACITY: ";
333     energy_upgrade_string            += std::to_string(this->storage_level * 200);
334     energy_upgrade_string            += " kWh\n";
335
336     energy_upgrade_string            += "LEVEL: ";
337     energy_upgrade_string            += std::to_string(this->storage_level);
338     energy_upgrade_string            += "\n\n";
339
340     if (this->storage_level < MAX_STORAGE_LEVELS) {
341         energy_upgrade_string        += "[D]: + 200 kWh (";
342         energy_upgrade_string        += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
343         energy_upgrade_string        += " K)\n";
344     }
345
346     else {
347         energy_upgrade_string += " * MAX LEVEL * \n";
348     }
349
350     sf::Text energy_upgrade_text = sf::Text(
351         energy_upgrade_string,
352         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
353         16
354     );
355
356     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
357     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
358     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
359
360     this->render_window_ptr->draw(energy_upgrade_text);
361
362     return;
363 } /* __drawUpgradeOptions() */

```

4.12.3.2 __handleKeyPressEvents()

```

void TidalTurbine::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

156 {
157     if (this->just_built) {
158         return;
159     }
160
161     switch (this->event_ptr->key.code) {
162         case (sf::Keyboard::U): {
163             this->__openUpgradeMenu();
164
165             break;
166         }
167
168
169         case (sf::Keyboard::W): {
170             if (this->production_menu_open) {
171                 //...
172             }
173
174             else if (this->upgrade_menu_open) {
175                 this->__upgradePowerCapacity();
176             }
177
178             break;
179         }
180
181
182         case (sf::Keyboard::S): {
183             //...
184
185             break;

```

```

186         }
187
188
189         case (sf::Keyboard::D): {
190             if (this->upgrade_menu_open) {
191                 this->__upgradeStorageCapacity();
192             }
193
194             break;
195         }
196
197
198         default: {
199             // do nothing!
200
201             break;
202         }
203     }
204
205     return;
206 } /* __handleKeyPressEvents() */

```

4.12.3.3 __handleMouseButtonEvents()

```

void TidalTurbine::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

221 {
222     if (this->just_built) {
223         return;
224     }
225
226     switch (this->event_ptr->mouseButton.button) {
227         case (sf::Mouse::Left): {
228             //...
229
230             break;
231         }
232
233
234         case (sf::Mouse::Right): {
235             //...
236
237             break;
238         }
239
240
241         default: {
242             // do nothing!
243
244             break;
245         }
246     }
247
248     return;
249 } /* __handleMouseButtonEvents() */

```

4.12.3.4 __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() */

```

4.12.3.5 __upgradePowerCapacity()

```

void TidalTurbine::__upgradePowerCapacity (
    void ) [private]

```

Helper method to upgrade power capacity.

```

114 {
115     if (this->credits < TIDAL_TURBINE_BUILD_COST) {
116         std::cout << "Cannot upgrade tidal turbine: insufficient credits (need "
117             << TIDAL_TURBINE_BUILD_COST << " K)" << std::endl;
118
119         this->__sendInsufficientCreditsMessage();
120         return;
121     }
122
123     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
124         return;
125     }
126
127     this->health = 100;
128
129     this->capacity_kW += 100;
130     this->upgrade_level++;
131
132     this->just_upgraded = true;
133
134     this->assets_manager_ptr->getSound("upgrade")->play();
135
136     this->__sendCreditsSpentMessage(TIDAL_TURBINE_BUILD_COST);
137     this->__sendTileStateRequest();
138     this->__sendGameStateRequest();
139
140     return;
141 } /* __upgradePowerCapacity() */

```

4.12.3.6 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](#).

```

556 {
557     // 1. if just built, call base method and return
558     if (this->just_built) {
559         TileImprovement::draw();
560
561         return;
562     }
563
564     // 2. handle upgrade effects
565     if (this->just_upgraded) {
566         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
567             this->tile_improvement_sprite_animated[i].setColor(
568                 sf::Color(
569                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
570                     255,
571                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
572                     255
573                 )
574             );
575
576             this->tile_improvement_sprite_animated[i].setScale(
577                 sf::Vector2f(
578                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
579                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
580                 )
581             );
582         }
583
584         this->upgrade_frame++;
585     }
586
587     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
588         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
589             this->tile_improvement_sprite_animated[i].setColor(
590                 sf::Color(255,255,255,255)
591             );
592
593             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
594         }
595
596         this->just_upgraded = false;
597         this->upgrade_frame = 0;
598     }
599
600     // 3. draw first element of animated sprite
601     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
602
603     // 4. draw second element of animated sprite
604     if (this->is_running) {
605         //...
606     }
607     else {
608         //...
609     }
610
611     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
612
613     // 5. draw storage upgrades
614     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
615         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
616     }
617
618     // 6. draw production menu
619     if (this->production_menu_open) {
620         this->render_window_ptr->draw(this->production_menu_backing);
621         this->render_window_ptr->draw(this->production_menu_backing_text);
622
623         //...
624     }
625
626     // 7. draw upgrade menu
627     if (this->upgrade_menu_open) {
628         this->render_window_ptr->draw(this->upgrade_menu_backing);
629         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
630
631         this->__drawUpgradeOptions();
632     }
633
634     this->frame++;
635 }

```

```

642     return;
643 } /* draw() */

```

4.12.3.7 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

462 {
463     //          32 char x 17 line console "-----\n";
464     std::string options_substring = "CAPACITY: ";
465     options_substring += std::to_string(this->capacity_kW);
466     options_substring += " kW (level ";
467     options_substring += std::to_string(this->upgrade_level);
468     options_substring += ")\n";
469
470     options_substring += "PRODUCTION: ";
471     options_substring += std::to_string(this->production_MWh);
472     options_substring += " MWh\n";
473
474     options_substring += "DISPATCHABLE: ";
475     options_substring += std::to_string(this->dispatchable_MWh);
476     options_substring += " MWh\n";
477
478     options_substring += "HEALTH: ";
479     options_substring += std::to_string(this->health);
480     options_substring += "/100\n";
481
482     options_substring += "\n";
483     options_substring += "**** TIDAL TURBINE OPTIONS ****\n";
484     options_substring += "\n";
485     options_substring += "      [E]: OPEN PRODUCTION MENU\n";
486     options_substring += "      [U]: OPEN UPGRADE MENU\n";
487     options_substring += "HOLD [P]: SCRAP (";
488     options_substring += std::to_string(SCRAP_COST);
489     options_substring += " K)";
490
491     return options_substring;
492 } /* getTileOptionsSubstring() */

```

4.12.3.8 processEvent()

```

void TidalTurbine::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

507 {
508     TileImprovement :: processEvent ();
509
510     if (this->event_ptr->type == sf::Event::KeyPressed) {
511         this->__handleKeyPressEvents();
512     }
513
514     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
515         this->__handleMouseButtonEvents();
516     }
517
518     return;
519 } /* processEvent() */

```

4.12.3.9 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
534 {
535     TileImprovement :: processMessage ();
536
537     //...
538
539     return;
540 } /* processMessage() */
```

4.12.4 Member Data Documentation

4.12.4.1 capacity_kW

```
int TidalTurbine::capacity_kW
```

The rated production capacity [kW] of the solar PV array.

4.12.4.2 dispatchable_MWh

```
int TidalTurbine::dispatchable_MWh
```

The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

4.12.4.3 production_MWh

```
int TidalTurbine::production_MWh
```

The current production [MWh] of the solar PV array.

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

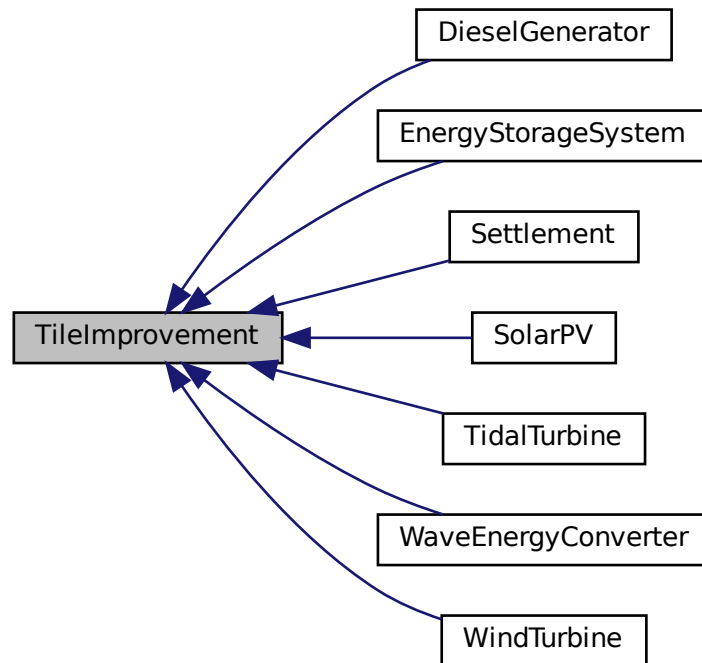
- header/[TidalTurbine.h](#)
- source/[TidalTurbine.cpp](#)

4.13 TileImprovement Class Reference

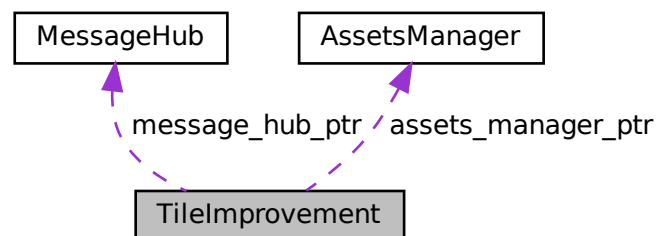
A base class for the tile improvement hierarchy.

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

Inheritance diagram for TileImprovement:



Collaboration diagram for TileImprovement:



Public Member Functions

- [TileImprovement](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [TileImprovement](#) class.
- virtual void [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 [just_upgraded](#)
A boolean which indicates that the improvement was just upgraded.
- bool [production_menu_open](#)
A boolean which indicates whether or not the production menu is open.
- bool [upgrade_menu_open](#)
A boolean which indicates whether or not the build 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.
- int [upgrade_level](#)
The upgrade level of the improvement.
- int [upgrade_frame](#)
The frame of the upgrade animation.
- int [storage_level](#)
The level of storage installed alongside the tile 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 menu.
- `sf::Text` [production_menu_backing_text](#)
Text for the production menu backing.
- `sf::RectangleShape` [upgrade_menu_backing](#)
A backing for the upgrade menu.
- `sf::Text` [upgrade_menu_backing_text](#)
Text for the upgrade menu backing.
- `sf::Sprite` [storage_upgrade_sprite](#)
A sprite for illustrating storage (in upgrade menu).
- `std::vector< sf::Sprite >` [storage_upgrade_sprite_vec](#)
A vector of sprites for illustrating the storage upgrade level (on tile).
- `sf::Sprite` [upgrade_arrow_sprite](#)
An upgrade arrow sprite.
- `sf::Sprite` [upgrade_plus_sprite](#)
An upgrade plus sprite.

Protected Member Functions

- `void` [__setUpProductionMenu](#) (void)
Helper method to set up and position production menu assets (drawable).
- `void` [__setUpUpgradeMenu](#) (void)
Helper method to set up and position upgrade menu assets (drawable).
- `void` [__upgradeStorageCapacity](#) (void)
Helper method to upgrade storage capacity.
- `void` [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- `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.
- `void` [__openUpgradeMenu](#) (void)
Helper method to open the upgrade menu.
- `void` [__closeUpgradeMenu](#) (void)
Helper method to close the build menu.
- `void` [__sendTileStateRequest](#) (void)
Helper method to format and send a request for the parent [HexTile](#) to send a tile state message.
- `void` [__sendGameStateRequest](#) (void)
Helper method to format and send a game state request (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.

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

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

```
565 {
566     // 1. set attributes
567
568     // 1.1. protected
569     this->event_ptr = event_ptr;
570     this->render_window_ptr = render_window_ptr;
571
572     this->assets_manager_ptr = assets_manager_ptr;
573     this->message_hub_ptr = message_hub_ptr;
574 }
```

```

575     // 1.2. public
576     this->is_selected = true;
577     this->just_built = true;
578     this->production_menu_open = false;
579     this->upgrade_menu_open = false;
580
581     this->frame = 0;
582     this->credits = 0;
583
584     this->position_x = position_x;
585     this->position_y = position_y;
586
587     this->game_phase = "build settlement";
588
589     this->__setUpProductionMenu();
590     this->__setUpUpgradeMenu();
591
592     std::cout << "TileImprovement constructed at " << this << std::endl;
593
594     return;
595 } /* TileImprovement() */

```

4.13.2.2 ~TileImprovement()

```

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

```

Destructor for the [TileImprovement](#) class.

```

813 {
814     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
815
816     return;
817 } /* ~TileImprovement() */

```

4.13.3 Member Function Documentation

4.13.3.1 __closeProductionMenu()

```

void TileImprovement::__closeProductionMenu (
    void ) [protected]

```

Helper method to close the production menu.

```

350 {
351     if (not this->production_menu_open) {
352         return;
353     }
354
355     this->production_menu_open = false;
356     this->assets_manager_ptr->getSound("build menu close")->play();
357
358     return;
359 } /* __closeProductionMenu() */

```

4.13.3.2 __closeUpgradeMenu()

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

Helper method to close the build menu.

```
402 {
403     if (not this->upgrade_menu_open) {
404         return;
405     }
406
407     this->upgrade_menu_open = false;
408     this->assets_manager_ptr->getSound("build menu close")->play();
409
410     return;
411 } /* __closeUpgradeMenu() */
```

4.13.3.3 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
235 {
236     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
237         return;
238     }
239
240     if (this->just_built) {
241         return;
242     }
243
244     switch (this->event_ptr->key.code) {
245         case (sf::Keyboard::E): {
246             this->__openProductionMenu();
247
248             break;
249         }
250
251         default: {
252             // do nothing!
253
254             break;
255         }
256     }
257
258     return;
259 } /* __handleKeyPressEvents() */
```

4.13.3.4 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
275 {
276     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
277         return;
278     }
279
280     if (this->just_built) {
281         return;
282     }
283
284     switch (this->event_ptr->mouseButton.button) {
```

```

285         case (sf::Mouse::Left): {
286             //...
287
288             break;
289         }
290
291         case (sf::Mouse::Right): {
292             //...
293
294             break;
295         }
296
297
298
299         default: {
300             // do nothing!
301
302             break;
303         }
304     }
305
306     return;
307 } /* __handleMouseButtonEvents() */

```

4.13.3.5 __openProductionMenu()

```

void TileImprovement::__openProductionMenu (
    void ) [protected]

```

Helper method to open the production menu.

```

322 {
323     if (this->production_menu_open) {
324         return;
325     }
326
327     if (this->upgrade_menu_open) {
328         this->__closeUpgradeMenu();
329     }
330
331     this->production_menu_open = true;
332     this->assets_manager_ptr->getSound("build menu open")->play();
333
334     return;
335 } /* __openProductionMenu() */

```

4.13.3.6 __openUpgradeMenu()

```

void TileImprovement::__openUpgradeMenu (
    void ) [protected]

```

Helper method to open the upgrade menu.

```

374 {
375     if (this->upgrade_menu_open) {
376         return;
377     }
378
379     if (this->production_menu_open) {
380         this->__closeProductionMenu();
381     }
382
383     this->upgrade_menu_open = true;
384     this->assets_manager_ptr->getSound("build menu open")->play();
385
386     return;
387 } /* __openUpgradeMenu() */

```

4.13.3.7 __sendCreditsSpentMessage()

```
void TileImprovement::__sendCreditsSpentMessage (
    int credits_spent ) [protected]
```

Helper method to format and send a credits spent message.

Parameters

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

```
479 {
480     Message credits_spent_message;
481
482     credits_spent_message.channel = GAME_CHANNEL;
483     credits_spent_message.subject = "credits spent";
484
485     credits_spent_message.int_payload["credits spent"] = credits_spent;
486
487     this->message_hub_ptr->sendMessage(credits_spent_message);
488
489     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
490         << std::endl;
491     return;
492 } /* __sendCreditsSpentMessage() */
```

4.13.3.8 __sendGameStateRequest()

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

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

```
452 {
453     Message game_state_request;
454
455     game_state_request.channel = GAME_CHANNEL;
456     game_state_request.subject = "state request";
457
458     this->message_hub_ptr->sendMessage(game_state_request);
459
460     std::cout << "Game state request message sent by " << this << std::endl;
461     return;
462 } /* __sendGameStateRequest() */
```

4.13.3.9 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
507 {
508     Message insufficient_credits_message;
509
510     insufficient_credits_message.channel = GAME_CHANNEL;
511     insufficient_credits_message.subject = "insufficient credits";
512
513     this->message_hub_ptr->sendMessage(insufficient_credits_message);
514
515     std::cout << "Insufficient credits message sent by " << this << std::endl;
516
517     return;
518 } /* __sendInsufficientCreditsMessage() */
```

4.13.3.10 __sendTileStateRequest()

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

Helper method to format and send a request for the parent [HexTile](#) to send a tile state message.

```
427 {
428     Message tile_state_request;
429
430     tile_state_request.channel = TILE_STATE_CHANNEL;
431     tile_state_request.subject = "state request";
432
433     this->message_hub_ptr->sendMessage(tile_state_request);
434
435     std::cout << "Tile state request sent by " << this << std::endl;
436     return;
437 } /* __sendTileStateRequest() */
```

4.13.3.11 __setUpProductionMenu()

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

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

```
68 {
69     // 1. set up and place production 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(MONOCHROME_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(MONOCHROME_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(400, 400 - 128 + 4);
87
88     return;
89 } /* __setUpProductionMenu() */
```

4.13.3.12 __setUpUpgradeMenu()

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

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

```
104 {
105     // 1. set up and place upgrade menu backing and text
106     this->upgrade_menu_backing.setSize(sf::Vector2f(400, 256));
107     this->upgrade_menu_backing.setOrigin(200, 128);
108     this->upgrade_menu_backing.setPosition(400, 400);
109     this->upgrade_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
110     this->upgrade_menu_backing.setOutlineColor(MENU_FRAME_GREY);
111     this->upgrade_menu_backing.setOutlineThickness(4);
112
113     this->upgrade_menu_backing_text.setString("**** UPGRADE MENU ****");
114     this->upgrade_menu_backing_text.setFont(
115         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
116     );
```



```

117     this->upgrade_menu_backing_text.setCharacterSize(16);
118     this->upgrade_menu_backing_text.setFillColor(MONOCROME_TEXT_GREEN);
119     this->upgrade_menu_backing_text.setOrigin(
120         this->upgrade_menu_backing_text.getLocalBounds().width / 2, 0
121     );
122     this->upgrade_menu_backing_text.setPosition(400, 400 - 128 + 4);
123
124
125     // 2. set up and place storage upgrade sprite (with upgrade plus)
126     this->storage_upgrade_sprite = sf::Sprite(
127         *(this->assets_manager_ptr->getTexture("energy storage system"))
128     );
129
130     this->storage_upgrade_sprite.setOrigin(
131         this->storage_upgrade_sprite.getLocalBounds().width / 2,
132         this->storage_upgrade_sprite.getLocalBounds().height
133     );
134
135     this->storage_upgrade_sprite.setPosition(400 + 100, 400 - 32);
136
137     this->upgrade_plus_sprite = sf::Sprite(
138         *(this->assets_manager_ptr->getTexture("upgrade plus"))
139     );
140
141     this->upgrade_plus_sprite.setOrigin(
142         this->upgrade_plus_sprite.getLocalBounds().width / 2,
143         this->upgrade_plus_sprite.getLocalBounds().height / 2
144     );
145
146     this->upgrade_plus_sprite.setPosition(400 + 130, 400 - 64);
147
148
149     // 3. set up and place upgrade arrow sprite
150     this->upgrade_arrow_sprite = sf::Sprite(
151         *(this->assets_manager_ptr->getTexture("upgrade arrow"))
152     );
153
154     this->upgrade_arrow_sprite.setOrigin(
155         this->upgrade_arrow_sprite.getLocalBounds().width / 2,
156         this->upgrade_arrow_sprite.getLocalBounds().height / 2
157     );
158
159     this->upgrade_arrow_sprite.setPosition(400 - 64, 400 - 64);
160
161
162     return;
163 } /* __setUpUpgradeMenu() */

```

4.13.3.13 __upgradeStorageCapacity()

```

void TileImprovement::__upgradeStorageCapacity (
    void ) [protected]

```

Helper method to upgrade storage capacity.

```

178 {
179     if (this->credits < ENERGY_STORAGE_SYSTEM_BUILD_COST) {
180         std::cout << "Cannot add energy storage: insufficient credits (need "
181             << ENERGY_STORAGE_SYSTEM_BUILD_COST << " K)" << std::endl;
182
183         this->__sendInsufficientCreditsMessage();
184         return;
185     }
186
187     if (this->storage_level >= MAX_STORAGE_LEVELS) {
188         return;
189     }
190
191     this->health = 100;
192
193     this->storage_level++;
194
195     this->storage_upgrade_sprite_vec.push_back(
196         sf::Sprite(
197             *(this->assets_manager_ptr->getTexture("storage level"))
198         )
199     );
200
201     this->storage_upgrade_sprite_vec.back().setOrigin(

```

```

202         this->storage_upgrade_sprite_vec.back().getLocalBounds().width / 2,
203         this->storage_upgrade_sprite_vec.back().getLocalBounds().height
204     );
205
206     this->storage_upgrade_sprite_vec.back().setPosition(
207         this->position_x + 18,
208         this->position_y + 25 - 7 * this->storage_upgrade_sprite_vec.size()
209     );
210
211     this->just_upgraded = true;
212
213     this->assets_manager_ptr->getSound("upgrade")->play();
214
215     this->__sendCreditsSpentMessage(ENERGY_STORAGE_SYSTEM_BUILD_COST);
216     this->__sendTileStateRequest();
217     this->__sendGameStateRequest();
218
219     return;
220 } /* __upgradeStorageCapacity() */

```

4.13.3.14 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](#).

```

684 {
685     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
686         int alpha = this->tile_improvement_sprite_static.getColor().a;
687
688         alpha += 0.08 * FRAMES_PER_SECOND;
689
690         this->tile_improvement_sprite_static.setColor(
691             sf::Color(255, 255, 255, alpha)
692         );
693
694         this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);
695
696         if (
697             (alpha >= 255) or
698             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
699         ) {
700             this->tile_improvement_sprite_static.setColor(
701                 sf::Color(255, 255, 255, 255)
702             );
703
704             this->tile_improvement_sprite_static.setPosition(
705                 this->position_x,
706                 this->position_y + 12
707             );
708
709             this->just_built = false;
710             this->assets_manager_ptr->getSound("place improvement")->play();
711         }
712
713         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
714     }
715
716     else {
717         int alpha = 0;
718
719         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
720             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
721
722             alpha += 0.08 * FRAMES_PER_SECOND;
723
724             this->tile_improvement_sprite_animated[i].setColor(
725                 sf::Color(255, 255, 255, alpha)
726             );
727
728             this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
729
730             if (
731

```

```

732         (alpha >= 255) or
733         (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
734     ) {
735         this->tile_improvement_sprite_animated[i].setColor(
736             sf::Color(255, 255, 255, 255)
737         );
738
739         this->tile_improvement_sprite_animated[i].setPosition(
740             this->position_x,
741             this->position_y + 12
742         );
743     }
744
745     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
746 }
747
748 if (
749     (alpha >= 255) or
750     (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
751 ) {
752     this->just_built = false;
753     this->assets_manager_ptr->getSound("place_improvement")->play();
754
755     switch (this->tile_improvement_type) {
756         case (TileImprovementType :: WIND_TURBINE): {
757             for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
758                 this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
759                 this->tile_improvement_sprite_animated[i].move(0, -32);
760             }
761
762             break;
763         }
764
765         case (TileImprovementType :: TIDAL_TURBINE): {
766             for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
767                 this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
768                 this->tile_improvement_sprite_animated[i].move(0, -19);
769             }
770
771             break;
772         }
773
774         case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
775             for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
776                 this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
777                 this->tile_improvement_sprite_animated[i].move(0, -32);
778             }
779
780             break;
781         }
782
783         default: {
784             // do nothing!
785
786             break;
787         }
788     }
789 }
790
791 }
792
793 }
794
795
796 this->frame++;
797 return;
798 } /* draw() */

```

4.13.3.15 getTileOptionsSubstring()

```

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

```

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

```
169 {return "";}

```

4.13.3.16 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](#).

```
639 {
640     if (this->event_ptr->type == sf::Event::KeyPressed) {
641         this->__handleKeyPressEvents();
642     }
643
644     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
645         this->__handleMouseButtonEvents();
646     }
647
648     return;
649 } /* processEvent() */
```

4.13.3.17 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](#).

```
664 {
665     //...
666
667     return;
668 } /* processMessage() */
```

4.13.3.18 setIsSelected()

```
void TileImprovement::setIsSelected (
    bool is_selected ) [virtual]
```

Method to set the is selected attribute.

Parameters

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

Reimplemented in [Settlement](#), and [EnergyStorageSystem](#).

```
612 {
613     this->is_selected = is_selected;
614
615     if ((not is_selected) and this->production_menu_open) {
616         this->__closeProductionMenu();
617     }
618
619     if ((not is_selected) and this->upgrade_menu_open) {
```

```
620         this->__closeUpgradeMenu();
621     }
622
623     return;
624 } /* 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 just_upgraded

```
bool TileImprovement::just_upgraded
```

A boolean which indicates that the improvement was just upgraded.

4.13.4.11 message_hub_ptr

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

A pointer to the message hub.

4.13.4.12 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.13 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.14 production_menu_backing

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

A backing for the production menu.

4.13.4.15 production_menu_backing_text

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

Text for the production menu backing.

4.13.4.16 production_menu_open

```
bool TileImprovement::production_menu_open
```

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

4.13.4.17 render_window_ptr

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

A pointer to the render window.

4.13.4.18 storage_level

```
int TileImprovement::storage_level
```

The level of storage installed alongside the tile improvement.

4.13.4.19 storage_upgrade_sprite

```
sf::Sprite TileImprovement::storage_upgrade_sprite
```

A sprite for illustrating storage (in upgrade menu).

4.13.4.20 storage_upgrade_sprite_vec

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

A vector of sprites for illustrating the storage upgrade level (on tile).

4.13.4.21 tile_improvement_sprite_animated

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

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

4.13.4.22 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.23 tile_improvement_string

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

A string representation of the tile improvement type.

4.13.4.24 tile_improvement_type

`TileImprovementType TileImprovement::tile_improvement_type`

The type of the tile improvement.

4.13.4.25 upgrade_arrow_sprite

`sf::Sprite TileImprovement::upgrade_arrow_sprite`

An upgrade arrow sprite.

4.13.4.26 upgrade_frame

`int TileImprovement::upgrade_frame`

The frame of the upgrade animation.

4.13.4.27 upgrade_level

`int TileImprovement::upgrade_level`

The upgrade level of the improvement.

4.13.4.28 upgrade_menu_backing

`sf::RectangleShape TileImprovement::upgrade_menu_backing`

A backing for the upgrade menu.

4.13.4.29 upgrade_menu_backing_text

`sf::Text TileImprovement::upgrade_menu_backing_text`

Text for the upgrade menu backing.

4.13.4.30 upgrade_menu_open

```
bool TileImprovement::upgrade_menu_open
```

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

4.13.4.31 upgrade_plus_sprite

```
sf::Sprite TileImprovement::upgrade_plus_sprite
```

An upgrade plus sprite.

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

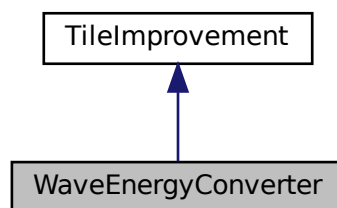
- header/[TileImprovement.h](#)
- source/[TileImprovement.cpp](#)

4.14 WaveEnergyConverter Class Reference

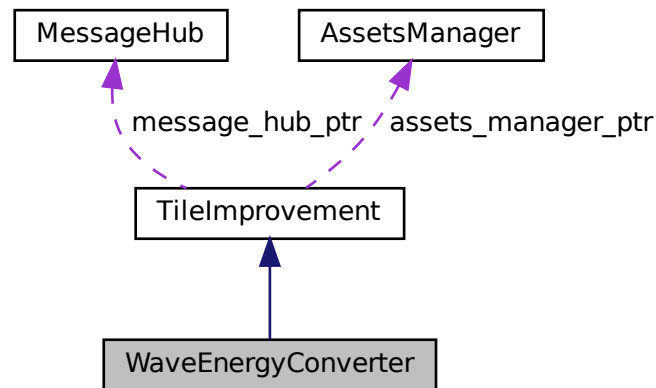
A settlement class (child class of [TileImprovement](#)).

```
#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.

Public Attributes

- int [capacity_kW](#)
The rated production capacity [kW] of the solar PV array.
- int [production_MWh](#)
The current production [MWh] of the solar PV array.
- int [dispatchable_MWh](#)
The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__upgradePowerCapacity](#) (void)
Helper method to upgrade power capacity.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__drawUpgradeOptions](#) (void)
Helper method to set up and draw upgrade options.

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.

```
408 :
409 TileImprovement (
```

```

410     position_x,
411     position_y,
412     event_ptr,
413     render_window_ptr,
414     assets_manager_ptr,
415     message_hub_ptr
416 )
417 {
418     // 1. set attributes
419
420     // 1.1. private
421     //...
422
423     // 1.2. public
424     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
425
426     this->is_running = false;
427
428     this->health = 100;
429
430     this->capacity_kW = 100;
431     this->upgrade_level = 1;
432     this->storage_level = 0;
433
434     this->production_MWh = 0;
435     this->dispatchable_MWh = 0;
436
437     this->tile_improvement_string = "WAVE ENERGY";
438
439     this->__setUpTileImprovementSpriteAnimated();
440
441     std::cout << "WaveEnergyConverter constructed at " << this << std::endl;
442
443     return;
444 } /* WaveEnergyConverter() */

```

4.14.2.2 ~WaveEnergyConverter()

```

WaveEnergyConverter::~WaveEnergyConverter (
    void ) [virtual]

```

Destructor for the [WaveEnergyConverter](#) class.

```

657 {
658     std::cout << "WaveEnergyConverter at " << this << " destroyed" << std::endl;
659
660     return;
661 } /* ~WaveEnergyConverter() */

```

4.14.3 Member Function Documentation

4.14.3.1 __drawUpgradeOptions()

```

void WaveEnergyConverter::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

263 {
264     // 1. draw power capacity upgrade sprite
265     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
266         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
267         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
268
269         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
270         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
271

```

```

272         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
273         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
274
275         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
276
277         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
278         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
279         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
280     }
281
282     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
283
284
285     // 2. draw power capacity upgrade text
286     //      16 char line = "                \n"
287     std::string power_upgrade_string = "POWER CAPACITY \n";
288     power_upgrade_string             += "                \n";
289
290     power_upgrade_string             += "CAPACITY: ";
291     power_upgrade_string             += std::to_string(this->capacity_kW);
292     power_upgrade_string             += " kW\n";
293
294     power_upgrade_string             += "LEVEL: ";
295     power_upgrade_string             += std::to_string(this->upgrade_level);
296     power_upgrade_string             += "\n\n";
297
298     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
299         power_upgrade_string         += "[W]: + 100 kW (";
300         power_upgrade_string         += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
301         power_upgrade_string         += " K)\n";
302     }
303
304     else {
305         power_upgrade_string         += " * MAX LEVEL * \n";
306     }
307
308     sf::Text power_upgrade_text = sf::Text(
309         power_upgrade_string,
310         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
311         16
312     );
313
314     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
315     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
316     power_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
317
318     this->render_window_ptr->draw(power_upgrade_text);
319
320
321     // 3. draw energy capacity (storage) upgrade sprite
322     this->render_window_ptr->draw(this->storage_upgrade_sprite);
323     this->render_window_ptr->draw(this->upgrade_plus_sprite);
324
325
326     // 4. draw energy capacity (storage) upgrade text
327     //      16 char line = "                \n"
328     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
329     energy_upgrade_string             += "                \n";
330
331     energy_upgrade_string             += "CAPACITY: ";
332     energy_upgrade_string             += std::to_string(this->storage_level * 200);
333     energy_upgrade_string             += " kWh\n";
334
335     energy_upgrade_string             += "LEVEL: ";
336     energy_upgrade_string             += std::to_string(this->storage_level);
337     energy_upgrade_string             += "\n\n";
338
339     if (this->storage_level < MAX_STORAGE_LEVELS) {
340         energy_upgrade_string         += "[D]: + 200 kWh (";
341         energy_upgrade_string         += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
342         energy_upgrade_string         += " K)\n";
343     }
344
345     else {
346         energy_upgrade_string += " * MAX LEVEL * \n";
347     }
348
349     sf::Text energy_upgrade_text = sf::Text(
350         energy_upgrade_string,
351         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
352         16
353     );
354
355     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
356     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
357     energy_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
358

```

```

359     this->render_window_ptr->draw(energy_upgrade_text);
360
361     return;
362 } /* __drawUpgradeOptions() */

```

4.14.3.2 __handleKeyPressEvents()

```

void WaveEnergyConverter::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

156 {
157     if (this->just_built) {
158         return;
159     }
160
161     switch (this->event_ptr->key.code) {
162         case (sf::Keyboard::U): {
163             this->__openUpgradeMenu();
164
165             break;
166         }
167
168         case (sf::Keyboard::W): {
169             if (this->production_menu_open) {
170                 //...
171             }
172
173             else if (this->upgrade_menu_open) {
174                 this->__upgradePowerCapacity();
175             }
176
177             break;
178         }
179
180         case (sf::Keyboard::S): {
181             //...
182
183             break;
184         }
185
186         case (sf::Keyboard::D): {
187             if (this->upgrade_menu_open) {
188                 this->__upgradeStorageCapacity();
189             }
190
191             break;
192         }
193
194         default: {
195             // do nothing!
196
197             break;
198         }
199     }
200
201     return;
202 } /* __handleKeyPressEvents() */

```

4.14.3.3 __handleMouseButtonEvents()

```

void WaveEnergyConverter::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

221 {
222     if (this->just_built) {
223         return;
224     }
225     switch (this->event_ptr->mouseButton.button) {
226         case (sf::Mouse::Left): {
227             //...
228
229             break;
230         }
231
232
233         case (sf::Mouse::Right): {
234             //...
235
236             break;
237         }
238
239
240         default: {
241             // do nothing!
242
243             break;
244         }
245     }
246
247     return;
248 } /* __handleMouseButtonEvents() */

```

4.14.3.4 __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() */

```


4.14.3.5 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

```
114 {
115     if (this->credits < WAVE_ENERGY_CONVERTER_BUILD_COST) {
116         std::cout << "Cannot upgrade wave energy converter: insufficient credits (need "
117             << WAVE_ENERGY_CONVERTER_BUILD_COST << " K)" << std::endl;
118
119         this->__sendInsufficientCreditsMessage();
120         return;
121     }
122
123     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
124         return;
125     }
126
127     this->health = 100;
128
129     this->capacity_kW += 100;
130     this->upgrade_level++;
131
132     this->just_upgraded = true;
133
134     this->assets_manager_ptr->getSound("upgrade")->play();
135
136     this->__sendCreditsSpentMessage(WAVE_ENERGY_CONVERTER_BUILD_COST);
137     this->__sendTileStateRequest();
138     this->__sendGameStateRequest();
139
140     return;
141 } /* __upgradePowerCapacity() */
```

4.14.3.6 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](#).

```
555 {
556     // 1. if just built, call base method and return
557     if (this->just_built) {
558         TileImprovement::draw();
559
560         return;
561     }
562
563     // 2. handle upgrade effects
564     if (this->just_upgraded) {
565         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
566             this->tile_improvement_sprite_animated[i].setColor(
567                 sf::Color(
568                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
569                     255,
570                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
571                     255
572                 )
573             );
574
575             this->tile_improvement_sprite_animated[i].setScale(
576                 sf::Vector2f(
577                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
578                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
579                 )
580             );
581         }
582
583         this->upgrade_frame++;
584     }
585 }
```

```

586
587     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
588         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
589             this->tile_improvement_sprite_animated[i].setColor(
590                 sf::Color(255,255,255,255)
591             );
592             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
593         }
594
595         this->just_upgraded = false;
596         this->upgrade_frame = 0;
597     }
598 }
599
600
601 // 3. draw first element of animated sprite
602 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
603
604
605 // 4. draw second element of animated sprite
606 if (this->is_running) {
607     //...
608 }
609
610 else {
611     //...
612 }
613
614 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
615
616
617 // 5. draw storage upgrades
618 for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
619     this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
620 }
621
622
623 // 6. draw production menu
624 if (this->production_menu_open) {
625     this->render_window_ptr->draw(this->production_menu_backing);
626     this->render_window_ptr->draw(this->production_menu_backing_text);
627
628     //...
629 }
630
631
632 // 7. draw upgrade menu
633 if (this->upgrade_menu_open) {
634     this->render_window_ptr->draw(this->upgrade_menu_backing);
635     this->render_window_ptr->draw(this->upgrade_menu_backing_text);
636
637     this->__drawUpgradeOptions();
638 }
639
640 this->frame++;
641 return;
642 } /* draw() */

```

4.14.3.7 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

461 {
462     // 32 char x 17 line console "-----\n";
463     std::string options_substring = "CAPACITY: ";
464     options_substring += std::to_string(this->capacity_kW);

```

```

465     options_substring      += " kW (level ";
466     options_substring      += std::to_string(this->upgrade_level);
467     options_substring      += ")\n";
468
469     options_substring      += "PRODUCTION: ";
470     options_substring      += std::to_string(this->production_MWh);
471     options_substring      += " MWh\n";
472
473     options_substring      += "DISPATCHABLE: ";
474     options_substring      += std::to_string(this->dispatchable_MWh);
475     options_substring      += " MWh\n";
476
477     options_substring      += "HEALTH: ";
478     options_substring      += std::to_string(this->health);
479     options_substring      += "/100\n";
480
481     options_substring      += " \n";
482     options_substring      += " **** WAVE ENERGY OPTIONS **** \n";
483     options_substring      += " \n";
484     options_substring      += "      [E]:  OPEN PRODUCTION MENU \n";
485     options_substring      += "      [U]:  OPEN UPGRADE MENU   \n";
486     options_substring      += "HOLD [P]:  SCRAP (";
487     options_substring      += std::to_string(SCRAP_COST);
488     options_substring      += " K)";
489
490     return options_substring;
491 } /* getTileOptionsSubstring() */

```

4.14.3.8 processEvent()

```

void WaveEnergyConverter::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

506 {
507     TileImprovement :: processEvent();
508
509     if (this->event_ptr->type == sf::Event::KeyPressed) {
510         this->__handleKeyPressEvents();
511     }
512
513     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
514         this->__handleMouseButtonEvents();
515     }
516
517     return;
518 } /* processEvent() */

```

4.14.3.9 processMessage()

```

void WaveEnergyConverter::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

533 {
534     TileImprovement :: processMessage();
535
536     //...
537
538     return;
539 } /* processMessage() */

```

4.14.4 Member Data Documentation

4.14.4.1 capacity_kW

```
int WaveEnergyConverter::capacity_kW
```

The rated production capacity [kW] of the solar PV array.

4.14.4.2 dispatchable_MWh

```
int WaveEnergyConverter::dispatchable_MWh
```

The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

4.14.4.3 production_MWh

```
int WaveEnergyConverter::production_MWh
```

The current production [MWh] of the solar PV array.

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

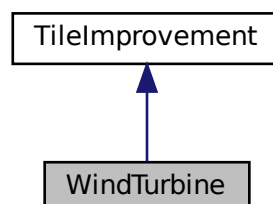
- header/[WaveEnergyConverter.h](#)
- source/[WaveEnergyConverter.cpp](#)

4.15 WindTurbine Class Reference

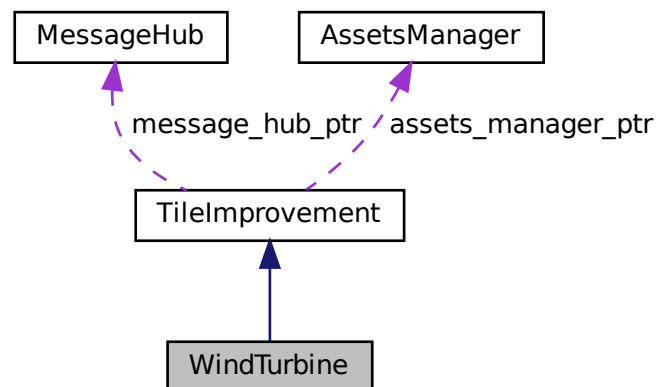
A settlement class (child class of [TileImprovement](#)).

```
#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.

Public Attributes

- int [capacity_kW](#)
The rated production capacity [kW] of the solar PV array.
- int [production_MWh](#)
The current production [MWh] of the solar PV array.
- int [dispatchable_MWh](#)
The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__upgradePowerCapacity](#) (void)
Helper method to upgrade the power capacity.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__drawUpgradeOptions](#) (void)
Helper method to set up and draw upgrade options.

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.

```
409 :
410 TileImprovement (
```

```

411     position_x,
412     position_y,
413     event_ptr,
414     render_window_ptr,
415     assets_manager_ptr,
416     message_hub_ptr
417 )
418 {
419     // 1. set attributes
420
421     // 1.1. private
422     //...
423
424     // 1.2. public
425     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
426
427     this->is_running = false;
428
429     this->health = 100;
430
431     this->capacity_kW = 100;
432     this->upgrade_level = 1;
433     this->storage_level = 0;
434
435     this->production_MWh = 0;
436     this->dispatchable_MWh = 0;
437
438     this->tile_improvement_string = "WIND TURBINE";
439
440     this->__setUpTileImprovementSpriteAnimated();
441
442     std::cout << "WindTurbine constructed at " << this << std::endl;
443
444     return;
445 } /* WindTurbine() */

```

4.15.2.2 ~WindTurbine()

```

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

```

Destructor for the [WindTurbine](#) class.

```

658 {
659     std::cout << "WindTurbine at " << this << " destroyed" << std::endl;
660
661     return;
662 } /* ~WindTurbine() */

```

4.15.3 Member Function Documentation

4.15.3.1 __drawUpgradeOptions()

```

void WindTurbine::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

264 {
265     // 1. draw power capacity upgrade sprite
266     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
267         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
268         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
269
270         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
271         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
272

```

```

273         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
274         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
275
276         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
277
278         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
279         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
280         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
281     }
282
283     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
284
285
286     // 2. draw power capacity upgrade text
287     //      16 char line = "                                \n"
288     std::string power_upgrade_string = "POWER CAPACITY \n";
289     power_upgrade_string += "                                \n";
290
291     power_upgrade_string += "CAPACITY: ";
292     power_upgrade_string += std::to_string(this->capacity_kW);
293     power_upgrade_string += " kW\n";
294
295     power_upgrade_string += "LEVEL: ";
296     power_upgrade_string += std::to_string(this->upgrade_level);
297     power_upgrade_string += "\n\n";
298
299     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
300         power_upgrade_string += "[W]: + 100 kW (";
301         power_upgrade_string += std::to_string(WIND_TURBINE_BUILD_COST);
302         power_upgrade_string += " K)\n";
303     }
304
305     else {
306         power_upgrade_string += " * MAX LEVEL * \n";
307     }
308
309     sf::Text power_upgrade_text = sf::Text(
310         power_upgrade_string,
311         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
312         16
313     );
314
315     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
316     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
317     power_upgrade_text.setFillColors(MONOCROME_TEXT_GREEN);
318
319     this->render_window_ptr->draw(power_upgrade_text);
320
321
322     // 3. draw energy capacity (storage) upgrade sprite
323     this->render_window_ptr->draw(this->storage_upgrade_sprite);
324     this->render_window_ptr->draw(this->upgrade_plus_sprite);
325
326
327     // 4. draw energy capacity (storage) upgrade text
328     //      16 char line = "                                \n"
329     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
330     energy_upgrade_string += "                                \n";
331
332     energy_upgrade_string += "CAPACITY: ";
333     energy_upgrade_string += std::to_string(this->storage_level * 200);
334     energy_upgrade_string += " kWh\n";
335
336     energy_upgrade_string += "LEVEL: ";
337     energy_upgrade_string += std::to_string(this->storage_level);
338     energy_upgrade_string += "\n\n";
339
340     if (this->storage_level < MAX_STORAGE_LEVELS) {
341         energy_upgrade_string += "[D]: + 200 kWh (";
342         energy_upgrade_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
343         energy_upgrade_string += " K)\n";
344     }
345
346     else {
347         energy_upgrade_string += " * MAX LEVEL * \n";
348     }
349
350     sf::Text energy_upgrade_text = sf::Text(
351         energy_upgrade_string,
352         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
353         16
354     );
355
356     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
357     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
358     energy_upgrade_text.setFillColors(MONOCROME_TEXT_GREEN);
359

```



```

360     this->render_window_ptr->draw(energy_upgrade_text);
361
362     return;
363 } /* __drawUpgradeOptions() */

```

4.15.3.2 __handleKeyPressEvents()

```

void WindTurbine::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

156 {
157     if (this->just_built) {
158         return;
159     }
160
161     switch (this->event_ptr->key.code) {
162         case (sf::Keyboard::U): {
163             this->__openUpgradeMenu();
164
165             break;
166         }
167
168         case (sf::Keyboard::W): {
169             if (this->production_menu_open) {
170                 //...
171             }
172
173             else if (this->upgrade_menu_open) {
174                 this->__upgradePowerCapacity();
175             }
176
177             break;
178         }
179
180         case (sf::Keyboard::S): {
181             //...
182
183             break;
184         }
185
186         case (sf::Keyboard::D): {
187             if (this->upgrade_menu_open) {
188                 this->__upgradeStorageCapacity();
189             }
190
191             break;
192         }
193
194         default: {
195             // do nothing!
196
197             break;
198         }
199     }
200
201     return;
202 } /* __handleKeyPressEvents() */

```

4.15.3.3 __handleMouseButtonEvents()

```

void WindTurbine::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

221 {
222     if (this->just_built) {
223         return;
224     }
225
226     switch (this->event_ptr->mouseButton.button) {
227         case (sf::Mouse::Left): {
228             //...
229
230             break;
231         }
232
233         case (sf::Mouse::Right): {
234             //...
235
236             break;
237         }
238
239         default: {
240             // do nothing!
241
242             break;
243         }
244     }
245
246     return;
247 }
248 /* __handleMouseButtonEvents() */
249 }

```

4.15.3.4 __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() */

```

4.15.3.5 __upgradePowerCapacity()

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

Helper method to upgrade the power capacity.

```
114 {
115     if (this->credits < WIND_TURBINE_BUILD_COST) {
116         std::cout << "Cannot upgrade wind turbine: insufficient credits (need "
117             << WIND_TURBINE_BUILD_COST << " K)" << std::endl;
118
119         this->__sendInsufficientCreditsMessage();
120         return;
121     }
122
123     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
124         return;
125     }
126
127     this->health = 100;
128
129     this->capacity_kW += 100;
130     this->upgrade_level++;
131
132     this->just_upgraded = true;
133
134     this->assets_manager_ptr->getSound("upgrade")->play();
135
136     this->__sendCreditsSpentMessage(WIND_TURBINE_BUILD_COST);
137     this->__sendTileStateRequest();
138     this->__sendGameStateRequest();
139
140     return;
141 } /* __upgradePowerCapacity() */
```

4.15.3.6 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](#).

```
556 {
557     // 1. if just built, call base method and return
558     if (this->just_built) {
559         TileImprovement::draw();
560
561         return;
562     }
563
564     // 2. handle upgrade effects
565     if (this->just_upgraded) {
566         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
567             this->tile_improvement_sprite_animated[i].setColor(
568                 sf::Color(
569                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
570                     255,
571                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
572                     255
573                 )
574             );
575
576             this->tile_improvement_sprite_animated[i].setScale(
577                 sf::Vector2f(
578                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
579                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
580                 )
581             );
582         }
583
584         this->upgrade_frame++;
585     }
586 }
```

```

587
588     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
589         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
590             this->tile_improvement_sprite_animated[i].setColor(
591                 sf::Color(255,255,255,255)
592             );
593
594             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
595         }
596
597         this->just_upgraded = false;
598         this->upgrade_frame = 0;
599     }
600
601
602     // 3. draw first element of animated sprite
603     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
604
605
606     // 4. draw second element of animated sprite
607     if (this->is_running) {
608         //...
609     }
610
611     else {
612         //...
613     }
614
615     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
616
617
618     // 5. draw storage upgrades
619     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
620         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
621     }
622
623
624     // 6. draw production menu
625     if (this->production_menu_open) {
626         this->render_window_ptr->draw(this->production_menu_backing);
627         this->render_window_ptr->draw(this->production_menu_backing_text);
628
629         //...
630     }
631
632
633     // 7. draw upgrade menu
634     if (this->upgrade_menu_open) {
635         this->render_window_ptr->draw(this->upgrade_menu_backing);
636         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
637
638         this->__drawUpgradeOptions();
639     }
640
641     this->frame++;
642     return;
643 } /* draw() */

```

4.15.3.7 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

462 {
463     // 32 char x 17 line console "-----\n";
464     std::string options_substring = "CAPACITY: ";
465     options_substring += std::to_string(this->capacity_kW);

```

```

466     options_substring      += " kW (level ";
467     options_substring      += std::to_string(this->upgrade_level);
468     options_substring      += ")\n";
469
470     options_substring      += "PRODUCTION: ";
471     options_substring      += std::to_string(this->production_MWh);
472     options_substring      += " MWh\n";
473
474     options_substring      += "DISPATCHABLE: ";
475     options_substring      += std::to_string(this->dispatchable_MWh);
476     options_substring      += " MWh\n";
477
478     options_substring      += "HEALTH: ";
479     options_substring      += std::to_string(this->health);
480     options_substring      += "/100\n";
481
482     options_substring      += " \n";
483     options_substring      += " **** WIND TURBINE OPTIONS **** \n";
484     options_substring      += " \n";
485     options_substring      += "      [E]: OPEN PRODUCTION MENU \n";
486     options_substring      += "      [U]: OPEN UPGRADE MENU \n";
487     options_substring      += "HOLD [P]: SCRAP (";
488     options_substring      += std::to_string(SCRAP_COST);
489     options_substring      += " K)";
490
491     return options_substring;
492 } /* getTileOptionsSubstring() */

```

4.15.3.8 processEvent()

```

void WindTurbine::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

507 {
508     TileImprovement :: processEvent ();
509
510     if (this->event_ptr->type == sf::Event::KeyPressed) {
511         this->__handleKeyPressEvents();
512     }
513
514     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
515         this->__handleMouseButtonEvents();
516     }
517
518     return;
519 } /* processEvent() */

```

4.15.3.9 processMessage()

```

void WindTurbine::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

534 {
535     TileImprovement :: processMessage ();
536
537     //...
538
539     return;
540 } /* processMessage() */

```

4.15.4 Member Data Documentation

4.15.4.1 capacity_kW

```
int WindTurbine::capacity_kW
```

The rated production capacity [kW] of the solar PV array.

4.15.4.2 dispatchable_MWh

```
int WindTurbine::dispatchable_MWh
```

The amount of production that is directly dispatchable to the grid (i.e. production correlated with demand).

4.15.4.3 production_MWh

```
int WindTurbine::production_MWh
```

The current production [MWh] of the solar PV array.

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

The diagram illustrates the relationship between header files and source files in a C++ project. It shows a hierarchy where source files include header files. Specifically, `source/Game.cpp` and `source/main.cpp` include `header/Game.h`. Then, `header/Game.h` and `source/ContextMenu.cpp` both include `header/ContextMenu.h`.

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.

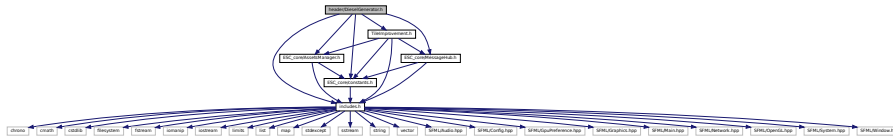
```

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

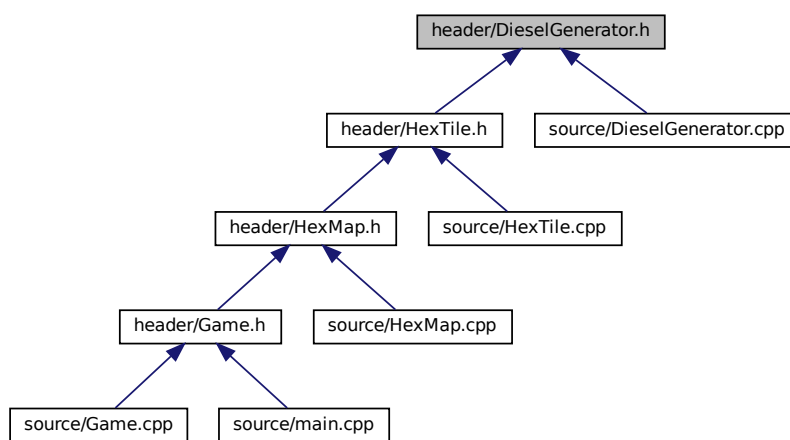
```



```
#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

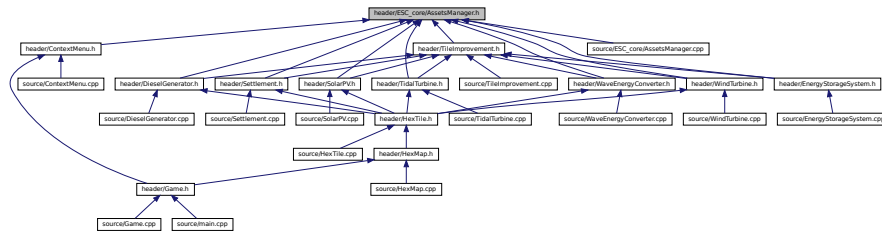
Header file for the [DieselGenerator](#) class.

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"
```


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



Classes

- class [AssetsManager](#)
A class which manages visual and sound assets.

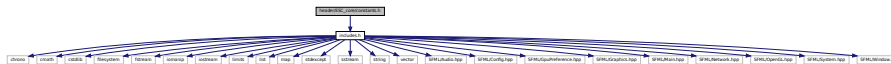
5.4.1 Detailed Description

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

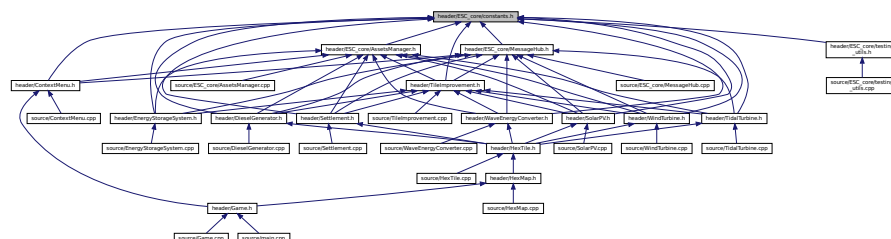
5.5 header/ESC_core/constants.h File Reference

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 in 100 kW increments.*
- const int `WIND_TURBINE_BUILD_COST` = 400
- The cost of building (or upgrading) a wind turbine in 100 kW increments.*
- 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 in 100 kW increments.*
- 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 in 100 kW increments.*
- const int `WAVE_ENERGY_CONVERTER_BUILD_COST` = 800
- The cost of building (or upgrading) a wave energy converter in 100 kW increments.*
- const int `ENERGY_STORAGE_SYSTEM_BUILD_COST` = 160
- The cost of adding energy storage in 200 kWh increments.*
- const int `SCRAP_COST` = 50
- The cost of scrapping a tile improvement (other than settlement).*
- const int `MAX_UPGRADE_LEVELS` = 5
- The maximum upgrade level of any tile improvement.*
- const int `MAX_STORAGE_LEVELS` = 5
- The maximum storage level of any tile improvement.*
- 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 in 100 kW increments.

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 = 160
```

The cost of adding energy storage in 200 kWh increments.

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 MAX_STORAGE_LEVELS

```
const int MAX_STORAGE_LEVELS = 5
```

The maximum storage level of any tile improvement.

5.5.3.17 MAX_UPGRADE_LEVELS

```
const int MAX_UPGRADE_LEVELS = 5
```

The maximum upgrade level of any tile improvement.

5.5.3.18 NO_TILE_SELECTED_CHANNEL

```
const std::string NO_TILE_SELECTED_CHANNEL = "NO TILE SELECTED CHANNEL"
```

A message channel for no tile selected messages.

5.5.3.19 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.20 SCRAP_COST

```
const int SCRAP_COST = 50
```

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

5.5.3.21 SECONDS_PER_FRAME

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

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

5.5.3.22 SECONDS_PER_MONTH

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

5.5.3.23 SECONDS_PER_YEAR

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

5.5.3.24 SOLAR_PV_BUILD_COST

```
const int SOLAR_PV_BUILD_COST = 300
```

The cost of building (or upgrading) a solar PV array in 100 kW increments.

5.5.3.25 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.26 STARTING_CREDITS

```
const int STARTING_CREDITS = 99999
```

The starting balance of credits.

5.5.3.27 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.28 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

The cost of building (or upgrading) a tidal turbine in 100 kW increments.

5.5.3.29 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).

5.5.3.30 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.31 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.32 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).

5.5.3.33 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800
```

The cost of building (or upgrading) a wave energy converter in 100 kW increments.

5.5.3.34 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

The cost of building (or upgrading) a wind turbine in 100 kW increments.

5.5.3.35 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

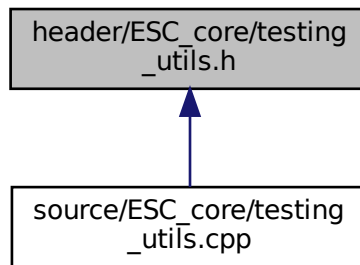
Header file which simply cites the doxygen tool.

5.6.1 Detailed Description

Header file which simply cites the doxygen tool.

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

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



Functions

- void `printGreen` (std::string)
A function that sends green text to std::cout.
- void `printGold` (std::string)
A function that sends gold text to std::cout.
- void `printRed` (std::string)
A function that sends red text to std::cout.
- void `testFloatEquals` (double, double, std::string, int)
Tests for the equality of two floating point numbers x and y (to within `FLOAT_TOLERANCE`).
- void `testGreaterThan` (double, double, std::string, int)
Tests if $x > y$.
- void `testGreaterThanOrEqualTo` (double, double, std::string, int)
Tests if $x \geq y$.
- void `testLessThan` (double, double, std::string, int)
Tests if $x < y$.
- void `testLessThanOrEqualTo` (double, double, std::string, int)
Tests if $x \leq y$.
- void `testTruth` (bool, std::string, int)
Tests if the given statement is true.
- void `expectedErrorNotDetected` (std::string, int)
A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

5.9.1 Detailed Description

Header file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

5.9.2 Function Documentation

5.9.2.1 expectedErrorNotDetected()

```
void expectedErrorNotDetected (
    std::string file,
    int line )
```

A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

Parameters

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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 << "\xB[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 std::cout.
------------------	---

```

134 {
135     std::cout << "\xB[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 "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

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 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    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: testGreaterThanOrEqualTo():\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 } /* testGreaterThanOrEqualTo() */

```

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.
----------	-----------------------------------

Parameters

<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() */

```

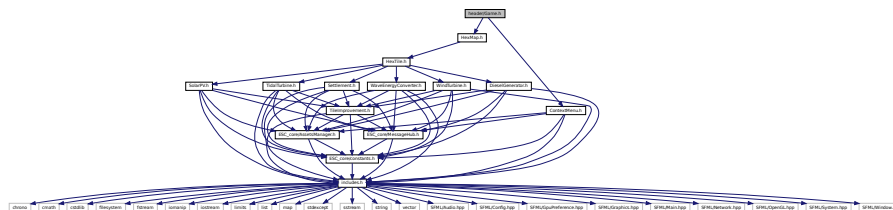
5.10 header/Game.h File Reference

```

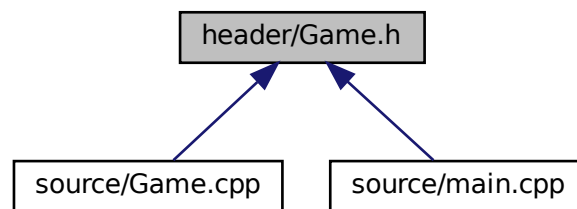
#include "HexMap.h"
#include "ContextMenu.h"

```

Include dependency graph for Game.h:



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



Classes

- class [Game](#)

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

Enumerations

- enum [GamePhase](#) {
[BUILD_SETTLEMENT](#) , [SYSTEM_MANAGEMENT](#) , [LOSS_EMISSIONS](#) , [LOSS_DEMAND](#) ,
[LOSS_CREDITS](#) , [VICTORY](#) , [N_GAME_PHASES](#) }

An enumeration of the various game phases.

5.10.1 Enumeration Type Documentation

5.10.1.1 GamePhase

enum [GamePhase](#)

An enumeration of the various game phases.

Enumerator

BUILD_SETTLEMENT	The settlement building phase.
SYSTEM_MANAGEMENT	The system management phase (main phase of play).
LOSS_EMISSIONS	A loss due to excessive emissions.
LOSS_DEMAND	A loss due to failing to meet the demand.
LOSS_CREDITS	A loss due to running out of credits.
VICTORY	A victory (12 consecutive months of zero emissions).
N_GAME_PHASES	A simple hack to get the number of elements in GamePhase.

```

66     {
67     BUILD\_SETTLEMENT,
68     SYSTEM\_MANAGEMENT,
69     LOSS\_EMISSIONS,
70     LOSS\_DEMAND,
71     LOSS\_CREDITS,
72     VICTORY,
73     N\_GAME\_PHASES
74 }; /* GamePhase */

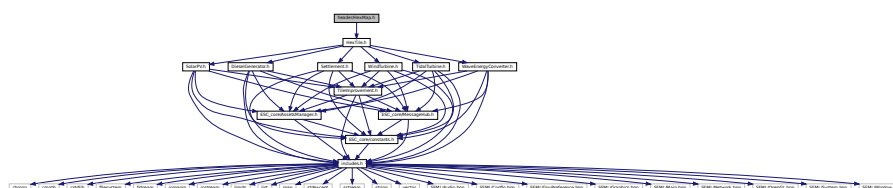
```

5.11 header/HexMap.h File Reference

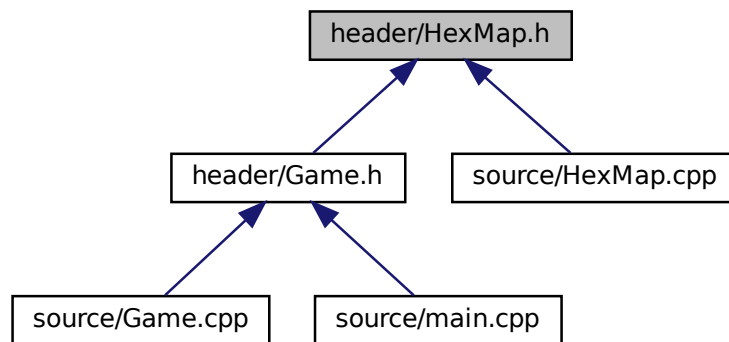
Header file for the [HexMap](#) class.

```
#include "HexTile.h"
```

Include dependency graph for HexMap.h:



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



Classes

- class [HexMap](#)

A class which defines a hex map of hex tiles.

5.11.1 Detailed Description

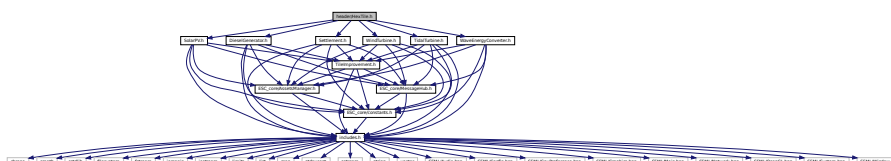
Header file for the [HexMap](#) class.

5.12 header/HexTile.h File Reference

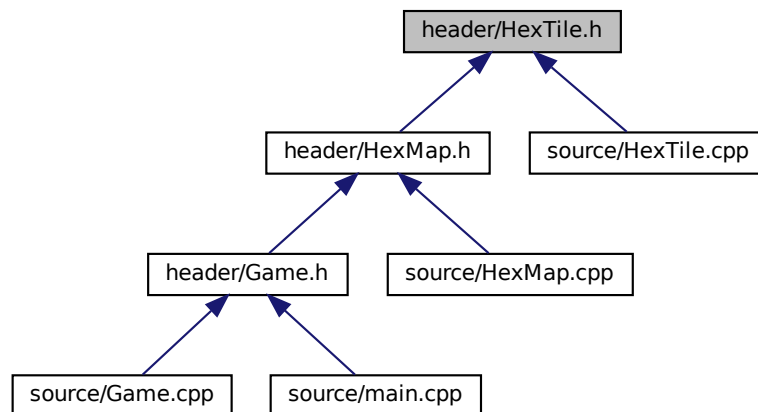
Header file for the [Game](#) class.

```
#include "DieselGenerator.h"
#include "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.

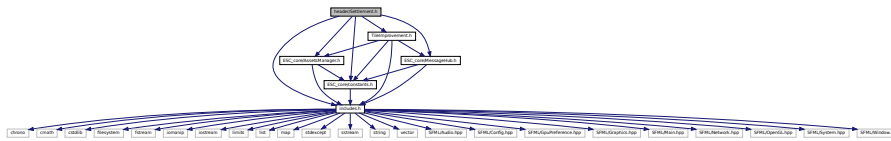
```

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

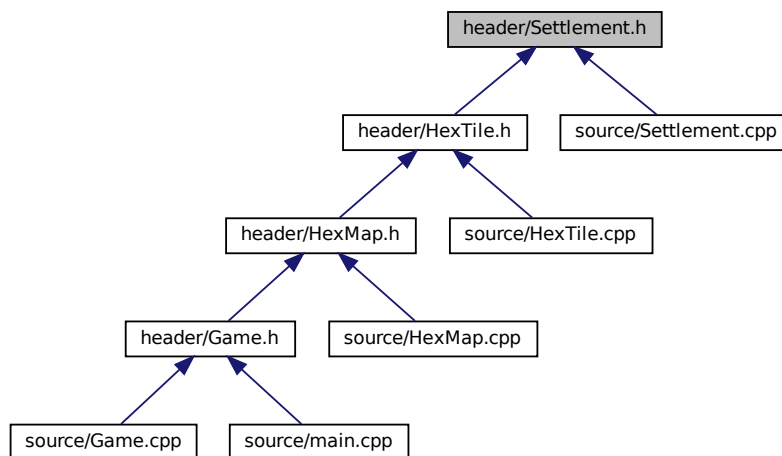
```

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

Include dependency graph for Settlement.h:



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



Classes

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

5.13.1 Detailed Description

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

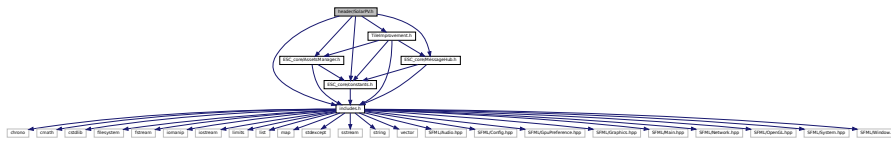
5.14 header/SolarPV.h File Reference

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

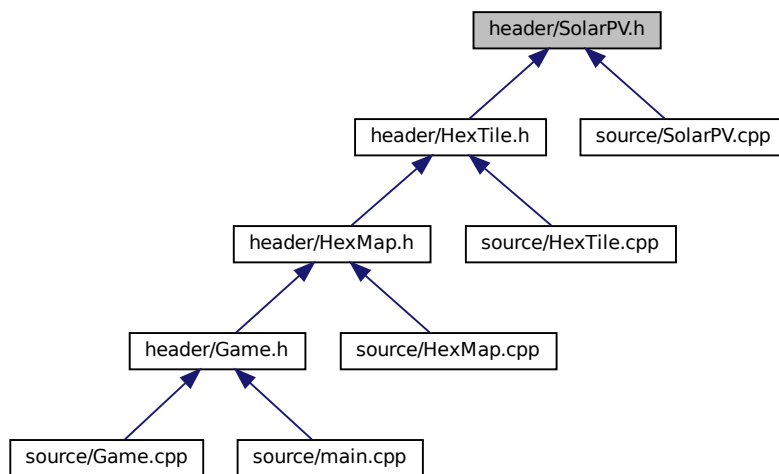
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

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

Include dependency graph for SolarPV.h:



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



Classes

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

5.14.1 Detailed Description

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

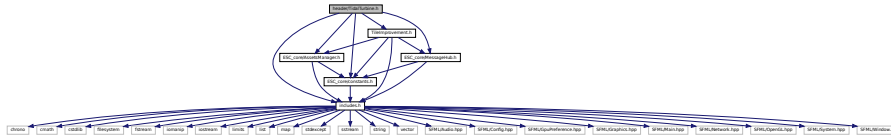
5.15 header/TidalTurbine.h File Reference

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

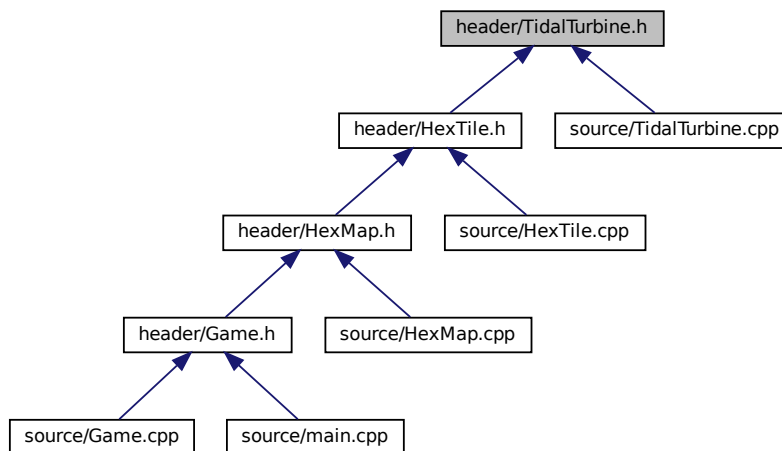
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

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

Include dependency graph for TidalTurbine.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

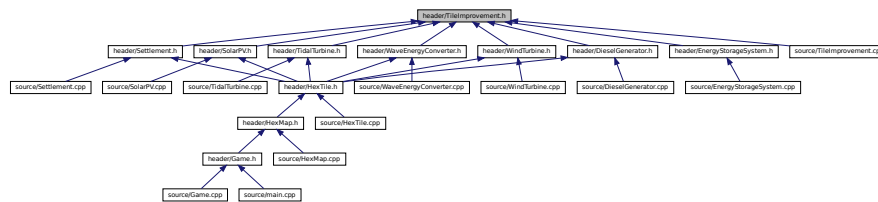
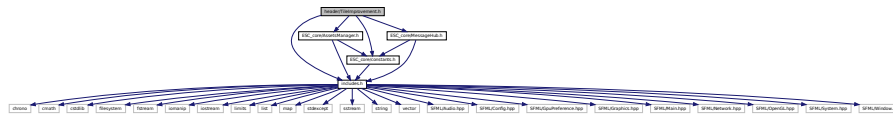
Header file for the [TidalTurbine](#) class.

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 dependency graph for TileImprovement.h:



- class `TileImprovement`
A base class for the tile improvement hierarchy.

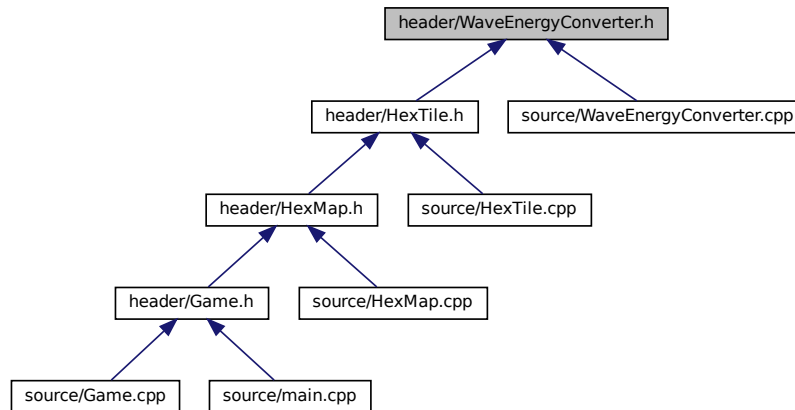
- `enum TileImprovementType {
SETTLEMENT, DIESEL_GENERATOR, SOLAR_PV, WIND_TURBINE,
TIDAL_TURBINE, WAVE_ENERGY_CONVERTER, ENERGY_STORAGE_SYSTEM, N_TILE_IMPROVEMENT_TYPES
}`
An enumeration of the different tile improvement types.

Header file for the `TileImprovement` class.

```
enum TileImprovementType
```

An enumeration of the different tile improvement types.

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



Classes

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

5.17.1 Detailed Description

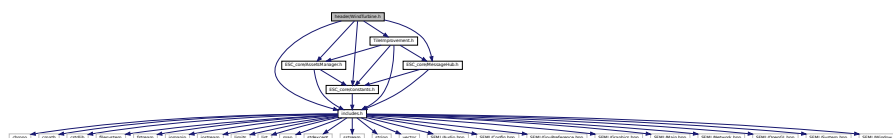
Header file for the [WaveEnergyConverter](#) class.

5.18 header/WindTurbine.h File Reference

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

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

Include dependency graph for `WindTurbine.h`:



Functions

- void `printGreen` (std::string input_str)
A function that sends green text to std::cout.
- void `printGold` (std::string input_str)
A function that sends gold text to std::cout.
- void `printRed` (std::string input_str)
A function that sends red text to std::cout.
- void `testFloatEquals` (double x, double y, std::string file, int line)
Tests for the equality of two floating point numbers x and y (to within FLOAT_TOLERANCE).
- void `testGreaterThan` (double x, double y, std::string file, int line)
Tests if $x > y$.
- void `testGreaterThanOrEqualTo` (double x, double y, std::string file, int line)
Tests if $x \geq y$.
- void `testLessThan` (double x, double y, std::string file, int line)
Tests if $x < y$.
- void `testLessThanOrEqualTo` (double x, double y, std::string file, int line)
Tests if $x \leq y$.
- void `testTruth` (bool statement, std::string file, int line)
Tests if the given statement is true.
- void `expectedErrorNotDetected` (std::string file, int line)
A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

5.24.1 Detailed Description

Implementation file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

5.24.2 Function Documentation

5.24.2.1 `expectedErrorNotDetected()`

```
void expectedErrorNotDetected (
    std::string file,
    int line )
```

A utility function to print out a meaningful error message whenever an expected error fails to be thrown/caught/detected.

Parameters

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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() */

```

5.25 source/Game.cpp File Reference

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

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     assets_manager_ptr->loadTexture(
149         "assets/tile_sheets/upgrade_arrow_16x16_1_CC-BY.png",
150         "upgrade arrow"
151     );
152
153     assets_manager_ptr->loadTexture(
154         "assets/tile_sheets/upgrade_plus_16x16_1_CC-BY.png",
155         "upgrade plus"
156     );
157
158     assets_manager_ptr->loadTexture(
159         "assets/tile_sheets/energy_storage_16x16_1_CC-BY.png",
160         "storage level"
161     );
162
163
164     // 3. load sounds
165     assets_manager_ptr->loadSound(
166         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
167         "coin ring"
168     );
169
170     assets_manager_ptr->loadSound(
171         "assets/audio/samples/mixkit-positive-notification-951_MixkitFree.ogg",
172         "positive notification"
173     );
174
175     assets_manager_ptr->loadSound(
176         "assets/audio/samples/mixkit-sci-fi-click-900_MixkitFree.ogg",
177         "sci-fi click"
178     );
179
180     assets_manager_ptr->loadSound(
181         "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932_MixkitFree.ogg",
182         "insufficient credits"
183     );
184
185     assets_manager_ptr->loadSound(
186         "assets/audio/samples/mixkit-data-scanner-2487_MixkitFree.ogg",
187         "resource assessment"
188     );
189
190     assets_manager_ptr->loadSound(
191         "assets/audio/samples/mixkit-interface-click-1126_MixkitFree.ogg",
192         "console string print"
193     );
194
195     assets_manager_ptr->loadSound(
196         "assets/audio/samples/mixkit-video-game-retro-click-237_MixkitFree.ogg",
197         "resource overlay toggle on"
198     );
199
200     assets_manager_ptr->loadSound(
201         "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED_MixkitFree.ogg",
202         "resource overlay toggle off"
203     );
204
205     assets_manager_ptr->loadSound(
206         "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703_MixkitFree.ogg",
207         "clear mountains tile"
208     );
209
210     assets_manager_ptr->loadSound(
211         "assets/audio/samples/mixkit-arcade-game-explosion-2759_MixkitFree.ogg",
212         "clear non-mountains tile"
213     );
214
215     assets_manager_ptr->loadSound(
216         "assets/audio/samples/mixkit-electronic-retro-block-hit-2185_MixkitFree.ogg",
217         "place improvement"
218     );
219
220     assets_manager_ptr->loadSound(
221         "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED_MixkitFree.ogg",
222         "build menu open"
```

```

223     );
224
225     assets_manager_ptr->loadSound(
226         "assets/audio/samples/mixkit-video-game-lock-2851_MixkitFree.ogg",
227         "build menu close"
228     );
229
230     assets_manager_ptr->loadSound(
231         "assets/audio/samples/mixkit-jump-into-the-water-1180_MixkitFree.ogg",
232         "splash"
233     );
234
235     assets_manager_ptr->loadSound(
236         "assets/audio/samples/505316__nuncaconoci__diesel_CC0.ogg",
237         "diesel running"
238     );
239
240     assets_manager_ptr->loadSound(
241         "assets/audio/samples/33460__pempi__320d_2_CC-BY.ogg",
242         "diesel start"
243     );
244
245     assets_manager_ptr->loadSound(
246         "assets/audio/samples/132724__andy_gardner__wind-turbine-blades_CC-BY.ogg",
247         "wind turbine running"
248     );
249
250     assets_manager_ptr->loadSound(
251         "assets/audio/samples/58416__darren1979__oceanwaves_CC-SAMPLING.ogg",
252         "ocean waves"
253     );
254
255     assets_manager_ptr->loadSound(
256         "assets/audio/samples/369927__mephisto_egmont__water-flowing-in-tubes_CC-BY.ogg",
257         "water flow"
258     );
259
260     assets_manager_ptr->loadSound(
261         "assets/audio/samples/647663__jotraing__electric-train-motor-idle-loop-new-generation-rollingstock_CC0.ogg",
262         "energy storage system"
263     );
264
265     assets_manager_ptr->loadSound(
266         "assets/audio/samples/mixkit-epic-futuristic-movie-accent-2913_MixkitFree.ogg",
267         "game title screen"
268     );
269
270     assets_manager_ptr->loadSound(
271         "assets/audio/samples/mixkit-calm-park-with-people-and-children_MixkitFree.ogg",
272         "people and children"
273     );
274
275     assets_manager_ptr->loadSound(
276         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
277         "upgrade"
278     );
279
280
281     // 4. load tracks
282     assets_manager_ptr->loadTrack(
283         "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
284         "Tree Star Moon - Dobranoc"
285     );
286
287     assets_manager_ptr->loadTrack(
288         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
289         "Tree Star Moon - Lighthouse"
290     );
291
292     assets_manager_ptr->loadTrack(
293         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
294         "Tree Star Moon - Sky Farm"
295     );
296
297     return;
298 } /* loadAssets() */

```

5.28.2.3 main()

```
int main (
```


5.30.1 Detailed Description

Implementation file for the **SolarPV** class.

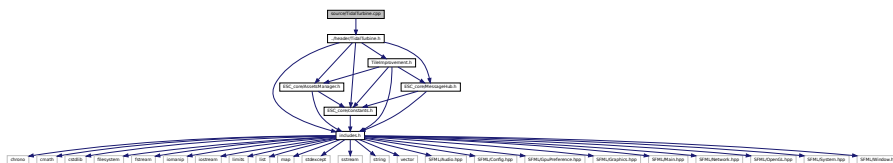
A base class for the tile improvement hierarchy.

5.31 source/TidalTurbine.cpp File Reference

Implementation file for the `TidalTurbine` class.

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

Include dependency graph for TidalTurbine.cpp:



5.31.1 Detailed Description

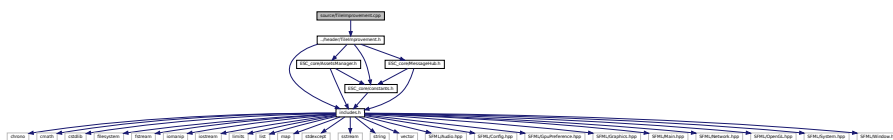
Implementation file for the `TidalTurbine` class.

A base class for the tile improvement hierarchy.

5.32 source/TileImprovement.cpp File Reference

Implementation file for the `TileImprovement` class.

```
#include "../header/TileImprovement.h"
Include dependency graph for TileImprovement.cpp:
```



5.32.1 Detailed Description

Implementation file for the `TileImprovement` 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/>. 229

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 228

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 39, 48, 97, 148, 156, 166, 177, 194, 204

Index

- __assembleHexMap
 - HexMap, [74](#)
- __assessNeighbours
 - HexMap, [74](#)
- __buildDieselGenerator
 - HexTile, [99](#)
- __buildDrawOrderVector
 - HexMap, [75](#)
- __buildEnergyStorage
 - HexTile, [99](#)
- __buildSettlement
 - HexTile, [100](#)
- __buildSolarPV
 - HexTile, [100](#)
- __buildTidalTurbine
 - HexTile, [101](#)
- __buildWaveEnergyConverter
 - HexTile, [101](#)
- __buildWindTurbine
 - HexTile, [102](#)
- __clearDecoration
 - HexTile, [103](#)
- __closeBuildMenu
 - HexTile, [103](#)
- __closeProductionMenu
 - TileImprovement, [178](#)
- __closeUpgradeMenu
 - TileImprovement, [178](#)
- __draw
 - Game, [58](#)
- __drawConsoleScreenFrame
 - ContextMenu, [22](#)
- __drawConsoleText
 - ContextMenu, [23](#)
- __drawFrameClockOverlay
 - Game, [58](#)
- __drawHUD
 - Game, [59](#)
- __drawUpgradeOptions
 - SolarPV, [157](#)
 - TidalTurbine, [167](#)
 - WaveEnergyConverter, [195](#)
 - WindTurbine, [205](#)
- __drawVisualScreenFrame
 - ContextMenu, [24](#)
- __enforceOceanContinuity
 - HexMap, [75](#)
- __getMajorityTileType
 - HexMap, [76](#)
- __getNeighboursVector
 - HexMap, [77](#)
- __getNoise
 - HexMap, [78](#)
- __getSelectedTile
 - HexMap, [79](#)
- __getTileCoordsSubstring
 - HexTile, [103](#)
- __getTileImprovementSubstring
 - HexTile, [104](#)
- __getTileOptionsSubstring
 - HexTile, [104](#)
- __getTileResourceSubstring
 - HexTile, [106](#)
- __getTileTypeSubstring
 - HexTile, [106](#)
- __getValidMapIndexPositions
 - HexMap, [80](#)
- __handleKeyPressEvents
 - ContextMenu, [24](#)
 - DieselGenerator, [40](#)
 - EnergyStorageSystem, [49](#)
 - Game, [60](#)
 - HexMap, [81](#)
 - HexTile, [107](#)
 - Settlement, [149](#)
 - SolarPV, [159](#)
 - TidalTurbine, [168](#)
 - TileImprovement, [179](#)
 - WaveEnergyConverter, [197](#)
 - WindTurbine, [207](#)
- __handleKeyReleaseEvents
 - HexTile, [111](#)
- __handleMouseButtonEvents
 - ContextMenu, [25](#)
 - DieselGenerator, [41](#)
 - EnergyStorageSystem, [50](#)
 - Game, [61](#)
 - HexMap, [81](#)
 - HexTile, [112](#)
 - Settlement, [149](#)
 - SolarPV, [159](#)
 - TidalTurbine, [169](#)
 - TileImprovement, [179](#)
 - WaveEnergyConverter, [197](#)
 - WindTurbine, [207](#)
- __insufficientCreditsAlarm
 - Game, [61](#)
- __isClicked

- HexTile, 112
- __isLakeTouchingOcean
 - HexMap, 82
- __layTiles
 - HexMap, 82
- __loadSoundBuffer
 - AssetsManager, 9
- __openBuildMenu
 - HexTile, 113
- __openProductionMenu
 - TileImprovement, 180
- __openUpgradeMenu
 - TileImprovement, 180
- __procedurallyGenerateTileResources
 - HexMap, 84
- __procedurallyGenerateTileTypes
 - HexMap, 85
- __processEvent
 - Game, 63
- __processMessage
 - Game, 63
- __scrapImprovement
 - HexTile, 113
- __sendAssessNeighboursMessage
 - HexTile, 114
- __sendCreditsSpentMessage
 - HexTile, 114
 - TileImprovement, 180
- __sendGameStateMessage
 - Game, 64
- __sendGameStateRequest
 - HexTile, 115
 - TileImprovement, 181
- __sendInsufficientCreditsMessage
 - HexTile, 115
 - TileImprovement, 181
- __sendNoTileSelectedMessage
 - HexMap, 85
- __sendQuitGameMessage
 - ContextMenu, 25
- __sendRestartGameMessage
 - ContextMenu, 25
- __sendTileSelectedMessage
 - HexTile, 115
- __sendTileStateMessage
 - HexTile, 116
- __sendTileStateRequest
 - TileImprovement, 181
- __sendUpdateGamePhaseMessage
 - HexTile, 116
- __setConsoleState
 - ContextMenu, 26
- __setConsoleString
 - ContextMenu, 26
- __setIsSelected
 - HexTile, 117
- __setResourceText
 - HexTile, 117
- __setUpBuildMenu
 - HexTile, 118
- __setUpBuildOption
 - HexTile, 119
- __setUpConsoleScreen
 - ContextMenu, 27
- __setUpConsoleScreenFrame
 - ContextMenu, 27
- __setUpDieselGeneratorBuildOption
 - HexTile, 120
- __setUpEnergyStorageSystemBuildOption
 - HexTile, 121
- __setUpGlassScreen
 - HexMap, 86
- __setUpMagnifyingGlassSprite
 - HexTile, 121
- __setUpMenuFrame
 - ContextMenu, 29
- __setUpNodeSprite
 - HexTile, 121
- __setUpProductionMenu
 - EnergyStorageSystem, 50
 - TileImprovement, 182
- __setUpResourceChipSprite
 - HexTile, 122
- __setUpSelectOutlineSprite
 - HexTile, 122
- __setUpSolarPVBuildOption
 - HexTile, 122
- __setUpTidalTurbineBuildOption
 - HexTile, 123
- __setUpTileExplosionReel
 - HexTile, 123
- __setUpTileImprovementSpriteAnimated
 - DieselGenerator, 41
 - TidalTurbine, 169
 - WaveEnergyConverter, 198
 - WindTurbine, 208
- __setUpTileImprovementSpriteStatic
 - EnergyStorageSystem, 51
 - Settlement, 150
 - SolarPV, 160
- __setUpTileSprite
 - HexTile, 124
- __setUpUpgradeMenu
 - TileImprovement, 182
- __setUpVisualScreen
 - ContextMenu, 30
- __setUpVisualScreenFrame
 - ContextMenu, 30
- __setUpWaveEnergyConverterBuildOption
 - HexTile, 124
- __setUpWindTurbineBuildOption
 - HexTile, 125
- __smoothTileTypes
 - HexMap, 86
- __toggleFrameClockOverlay
 - Game, 65

- __upgrade
 - DieselGenerator, 42
 - EnergyStorageSystem, 51
- __upgradePowerCapacity
 - SolarPV, 160
 - TidalTurbine, 170
 - WaveEnergyConverter, 198
 - WindTurbine, 208
- __upgradeStorageCapacity
 - TileImprovement, 183
- ~AssetsManager
 - AssetsManager, 8
- ~ContextMenu
 - ContextMenu, 22
- ~DieselGenerator
 - DieselGenerator, 40
- ~EnergyStorageSystem
 - EnergyStorageSystem, 49
- ~Game
 - Game, 58
- ~HexMap
 - HexMap, 74
- ~HexTile
 - HexTile, 98
- ~MessageHub
 - MessageHub, 141
- ~Settlement
 - Settlement, 149
- ~SolarPV
 - SolarPV, 157
- ~TidalTurbine
 - TidalTurbine, 166
- ~TileImprovement
 - TileImprovement, 178
- ~WaveEnergyConverter
 - WaveEnergyConverter, 195
- ~WindTurbine
 - WindTurbine, 205
- ABOVE_AVERAGE
 - HexTile.h, 241
- addChannel
 - MessageHub, 141
- assess
 - HexMap, 86
 - HexTile, 125
- assets_manager_ptr
 - ContextMenu, 33
 - Game, 66
 - HexMap, 90
 - HexTile, 132
 - TileImprovement, 187
- 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, 241
- BELOW_AVERAGE
 - HexTile.h, 241
- bool_payload
 - Message, 139
- border_tiles_vec
 - HexMap, 90
- build_menu_backing
 - HexTile, 132
- build_menu_backing_text
 - HexTile, 132
- build_menu_open
 - HexTile, 133
- build_menu_options_text_vec
 - HexTile, 133
- build_menu_options_vec
 - HexTile, 133
- BUILD_SETTLEMENT
 - Game.h, 238
- BUILD_SETTLEMENT_COST
 - constants.h, 222
- capacity_kW
 - DieselGenerator, 45
 - SolarPV, 163
 - TidalTurbine, 173
 - WaveEnergyConverter, 202
 - WindTurbine, 212
- capacity_MWh
 - EnergyStorageSystem, 54
- channel
 - Message, 139
- charge_MWh
 - EnergyStorageSystem, 54
- clear
 - AssetsManager, 10
 - HexMap, 87
 - MessageHub, 142

- CLEAR_FOREST_COST
 - constants.h, [222](#)
- CLEAR_MOUNTAINS_COST
 - constants.h, [222](#)
- CLEAR_PLAINS_COST
 - constants.h, [223](#)
- clearMessages
 - MessageHub, [142](#)
- clock
 - Game, [67](#)
- CO2E_KG_PER_LITRE_DIESEL
 - constants.h, [223](#)
- 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, [214](#)
- constants.h
 - BUILD_SETTLEMENT_COST, [222](#)
 - CLEAR_FOREST_COST, [222](#)
 - CLEAR_MOUNTAINS_COST, [222](#)
 - CLEAR_PLAINS_COST, [223](#)
 - CO2E_KG_PER_LITRE_DIESEL, [223](#)
 - DIESEL_GENERATOR_BUILD_COST, [223](#)
 - EMISSIONS_LIFETIME_LIMIT_TONNES, [223](#)
 - ENERGY_STORAGE_SYSTEM_BUILD_COST, [223](#)
 - FLOAT_TOLERANCE, [223](#)
 - FOREST_GREEN, [220](#)
 - FRAMES_PER_SECOND, [224](#)
 - GAME_CHANNEL, [224](#)
 - GAME_HEIGHT, [224](#)
 - GAME_STATE_CHANNEL, [224](#)
 - GAME_WIDTH, [224](#)
 - HEX_MAP_CHANNEL, [224](#)
 - LAKE_BLUE, [220](#)
 - MAX_STORAGE_LEVELS, [225](#)
 - MAX_UPGRADE_LEVELS, [225](#)
 - MENU_FRAME_GREY, [220](#)
 - MONOCHROME_SCREEN_BACKGROUND, [220](#)
 - MONOCHROME_TEXT_AMBER, [220](#)
 - MONOCHROME_TEXT_GREEN, [221](#)
 - MONOCHROME_TEXT_RED, [221](#)
 - MOUNTAINS_GREY, [221](#)
 - NO_TILE_SELECTED_CHANNEL, [225](#)
 - OCEAN_BLUE, [221](#)
 - PLAINS_YELLOW, [221](#)
 - RESOURCE_ASSESSMENT_COST, [225](#)
 - RESOURCE_CHIP_GREY, [222](#)
 - SCRAP_COST, [225](#)
 - SECONDS_PER_FRAME, [225](#)
 - SECONDS_PER_MONTH, [226](#)
 - SECONDS_PER_YEAR, [226](#)
 - SOLAR_PV_BUILD_COST, [226](#)
 - SOLAR_PV_WATER_BUILD_MULTIPLIER, [226](#)
 - STARTING_CREDITS, [226](#)
 - STARTING_POPULATION, [226](#)
 - TIDAL_TURBINE_BUILD_COST, [227](#)
 - TILE_RESOURCE_CUMULATIVE_PROBABILITIES, [227](#)
 - TILE_SELECTED_CHANNEL, [227](#)
 - TILE_STATE_CHANNEL, [227](#)
 - TILE_TYPE_CUMULATIVE_PROBABILITIES, [227](#)
 - VISUAL_SCREEN_FRAME_GREY, [222](#)
 - WAVE_ENERGY_CONVERTER_BUILD_COST, [228](#)
 - WIND_TURBINE_BUILD_COST, [228](#)
 - WIND_TURBINE_WATER_BUILD_MULTIPLIER, [228](#)
- constructRenderWindow
 - main.cpp, [258](#)
- context_menu_ptr
 - Game, [67](#)
- 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
- ContextMenu.h
 - ConsoleState, 214
 - MENU, 214
 - N_CONSOLE_STATES, 214
 - NONE_STATE, 214
 - READY, 214
 - TILE, 214
- credits
 - Game, 67
 - HexTile, 133
 - TileImprovement, 187
- cumulative_emissions_tonnes
 - Game, 67
- current_track
 - AssetsManager, 18
- decorateTile
 - HexTile, 126
- decoration_cleared
 - HexTile, 133
- demand_MWh
 - Game, 67
- DIESEL_GENERATOR
 - TileImprovement.h, 246
- DIESEL_GENERATOR_BUILD_COST
 - constants.h, 223
- DieselGenerator, 37
 - __handleKeyPressEvents, 40
 - __handleMouseButtonEvents, 41
 - __setUpTileImprovementSpriteAnimated, 41
 - __upgrade, 42
 - ~DieselGenerator, 40
 - capacity_kW, 45
 - DieselGenerator, 39
 - draw, 42
 - getTileOptionsSubstring, 43
 - max_production_MWh, 45
 - processEvent, 44
 - processMessage, 44
 - production_MWh, 45
 - smoke_da, 45
 - smoke_dx, 45
 - smoke_dy, 46
 - smoke_prob, 46
 - smoke_sprite_list, 46
- dispatchable_MWh
 - SolarPV, 163
 - TidalTurbine, 173
 - WaveEnergyConverter, 202
 - WindTurbine, 212
- double_payload
 - Message, 139
- draw
 - ContextMenu, 31
 - DieselGenerator, 42
 - EnergyStorageSystem, 52
 - HexMap, 87
 - HexTile, 127
 - Settlement, 150
 - SolarPV, 161
 - TidalTurbine, 170
 - TileImprovement, 184
 - WaveEnergyConverter, 199
 - WindTurbine, 209
- draw_explosion
 - HexTile, 133
- EMISSIONS_LIFETIME_LIMIT_TONNES
 - constants.h, 223
- ENERGY_STORAGE_SYSTEM
 - TileImprovement.h, 246
- ENERGY_STORAGE_SYSTEM_BUILD_COST
 - constants.h, 223
- EnergyStorageSystem, 47
 - __handleKeyPressEvents, 49
 - __handleMouseButtonEvents, 50
 - __setUpProductionMenu, 50
 - __setUpTileImprovementSpriteStatic, 51
 - __upgrade, 51
 - ~EnergyStorageSystem, 49
 - capacity_MWh, 54
 - charge_MWh, 54
 - draw, 52
 - EnergyStorageSystem, 48
 - getTileOptionsSubstring, 52
 - processEvent, 53
 - processMessage, 53
 - setIsSelected, 54
- event
 - Game, 67
- event_ptr
 - ContextMenu, 35
 - HexMap, 90
 - HexTile, 134
 - TileImprovement, 187
- expectedErrorNotDetected
 - testing_utils.cpp, 251
 - testing_utils.h, 231
- explosion_frame
 - HexTile, 134
- explosion_sprite_reel
 - HexTile, 134

- FLOAT_TOLERANCE
 - constants.h, [223](#)
- font_map
 - AssetsManager, [18](#)
- FOREST
 - HexTile.h, [241](#)
- FOREST_GREEN
 - constants.h, [220](#)
- frame
 - ContextMenu, [35](#)
 - Game, [68](#)
 - HexMap, [90](#)
 - HexTile, [134](#)
 - TileImprovement, [187](#)
- FRAMES_PER_SECOND
 - constants.h, [224](#)
- Game, [55](#)
 - __draw, [58](#)
 - __drawFrameClockOverlay, [58](#)
 - __drawHUD, [59](#)
 - __handleKeyPressEvents, [60](#)
 - __handleMouseButtonEvents, [61](#)
 - __insufficientCreditsAlarm, [61](#)
 - __processEvent, [63](#)
 - __processMessage, [63](#)
 - __sendGameStateMessage, [64](#)
 - __toggleFrameClockOverlay, [65](#)
 - ~Game, [58](#)
 - assets_manager_ptr, [66](#)
 - clock, [67](#)
 - context_menu_ptr, [67](#)
 - credits, [67](#)
 - cumulative_emissions_tonnes, [67](#)
 - demand_MWh, [67](#)
 - event, [67](#)
 - frame, [68](#)
 - Game, [57](#)
 - game_loop_broken, [68](#)
 - game_phase, [68](#)
 - hex_map_ptr, [68](#)
 - message_hub, [68](#)
 - month, [68](#)
 - population, [69](#)
 - quit_game, [69](#)
 - render_window_ptr, [69](#)
 - run, [66](#)
 - show_frame_clock_overlay, [69](#)
 - time_since_start_s, [69](#)
 - turn, [69](#)
 - year, [70](#)
- Game.h
 - BUILD_SETTLEMENT, [238](#)
 - GamePhase, [238](#)
 - LOSS_CREDITS, [238](#)
 - LOSS_DEMAND, [238](#)
 - LOSS_EMISSIONS, [238](#)
 - N_GAME_PHASES, [238](#)
 - SYSTEM_MANAGEMENT, [238](#)
 - VICTORY, [238](#)
- GAME_CHANNEL
 - constants.h, [224](#)
- GAME_HEIGHT
 - constants.h, [224](#)
- game_loop_broken
 - Game, [68](#)
- game_menu_up
 - ContextMenu, [35](#)
- game_phase
 - Game, [68](#)
 - HexTile, [134](#)
 - TileImprovement, [187](#)
- GAME_STATE_CHANNEL
 - constants.h, [224](#)
- GAME_WIDTH
 - constants.h, [224](#)
- GamePhase
 - Game.h, [238](#)
- getCurrentTrackKey
 - AssetsManager, [11](#)
- getFont
 - AssetsManager, [11](#)
- getSound
 - AssetsManager, [12](#)
- getSoundBuffer
 - AssetsManager, [12](#)
- getTexture
 - AssetsManager, [13](#)
- getTileOptionsSubstring
 - DieselGenerator, [43](#)
 - EnergyStorageSystem, [52](#)
 - Settlement, [151](#)
 - SolarPV, [162](#)
 - TidalTurbine, [172](#)
 - TileImprovement, [185](#)
 - WaveEnergyConverter, [200](#)
 - WindTurbine, [210](#)
- getTrackStatus
 - AssetsManager, [13](#)
- glass_screen
 - HexMap, [90](#)
- GOOD
 - HexTile.h, [241](#)
- has_improvement
 - HexTile, [134](#)
- hasTraffic
 - MessageHub, [142](#)
- header/ContextMenu.h, [213](#)
- header/DieselGenerator.h, [214](#)
- header/EnergyStorageSystem.h, [215](#)
- header/ESC_core/AssetsManager.h, [216](#)
- header/ESC_core/constants.h, [217](#)
- header/ESC_core/doxygen_cite.h, [228](#)
- header/ESC_core/includes.h, [229](#)
- header/ESC_core/MessageHub.h, [230](#)
- header/ESC_core/testing_utils.h, [230](#)
- header/Game.h, [237](#)

- header/HexMap.h, 238
- header/HexTile.h, 239
- header/Settlement.h, 241
- header/SolarPV.h, 242
- header/TidalTurbine.h, 243
- header/TileImprovement.h, 244
- header/WaveEnergyConverter.h, 246
- header/WindTurbine.h, 247
- health
 - TileImprovement, 187
- hex_draw_order_vec
 - HexMap, 91
- hex_map
 - HexMap, 91
- HEX_MAP_CHANNEL
 - constants.h, 224
- hex_map_ptr
 - Game, 68
- HexMap, 70
 - __assembleHexMap, 74
 - __assessNeighbours, 74
 - __buildDrawOrderVector, 75
 - __enforceOceanContinuity, 75
 - __getMajorityTileType, 76
 - __getNeighboursVector, 77
 - __getNoise, 78
 - __getSelectedTile, 79
 - __getValidMapIndexPositions, 80
 - __handleKeyPressEvents, 81
 - __handleMouseButtonEvents, 81
 - __isLakeTouchingOcean, 82
 - __layTiles, 82
 - __procedurallyGenerateTileResources, 84
 - __procedurallyGenerateTileTypes, 85
 - __sendNoTileSelectedMessage, 85
 - __setUpGlassScreen, 86
 - __smoothTileTypes, 86
 - ~HexMap, 74
 - assess, 86
 - assets_manager_ptr, 90
 - border_tiles_vec, 90
 - clear, 87
 - draw, 87
 - event_ptr, 90
 - frame, 90
 - glass_screen, 90
 - hex_draw_order_vec, 91
 - hex_map, 91
 - HexMap, 73
 - message_hub_ptr, 91
 - n_layers, 91
 - n_tiles, 91
 - position_x, 91
 - position_y, 92
 - processEvent, 88
 - processMessage, 88
 - render_window_ptr, 92
 - reroll, 89
 - show_resource, 92
 - tile_position_x_vec, 92
 - tile_position_y_vec, 92
 - tile_selected, 92
 - toggleResourceOverlay, 89
- HexTile, 93
 - __buildDieselGenerator, 99
 - __buildEnergyStorage, 99
 - __buildSettlement, 100
 - __buildSolarPV, 100
 - __buildTidalTurbine, 101
 - __buildWaveEnergyConverter, 101
 - __buildWindTurbine, 102
 - __clearDecoration, 103
 - __closeBuildMenu, 103
 - __getTileCoordsSubstring, 103
 - __getTileImprovementSubstring, 104
 - __getTileOptionsSubstring, 104
 - __getTileResourceSubstring, 106
 - __getTileTypeSubstring, 106
 - __handleKeyPressEvents, 107
 - __handleKeyReleaseEvents, 111
 - __handleMouseButtonEvents, 112
 - __isClicked, 112
 - __openBuildMenu, 113
 - __scrapImprovement, 113
 - __sendAssessNeighboursMessage, 114
 - __sendCreditsSpentMessage, 114
 - __sendGameStateRequest, 115
 - __sendInsufficientCreditsMessage, 115
 - __sendTileSelectedMessage, 115
 - __sendTileStateMessage, 116
 - __sendUpdateGamePhaseMessage, 116
 - __setIsSelected, 117
 - __setResourceText, 117
 - __setUpBuildMenu, 118
 - __setUpBuildOption, 119
 - __setUpDieselGeneratorBuildOption, 120
 - __setUpEnergyStorageSystemBuildOption, 121
 - __setUpMagnifyingGlassSprite, 121
 - __setUpNodeSprite, 121
 - __setUpResourceChipSprite, 122
 - __setUpSelectOutlineSprite, 122
 - __setUpSolarPVBuildOption, 122
 - __setUpTidalTurbineBuildOption, 123
 - __setUpTileExplosionReel, 123
 - __setUpTileSprite, 124
 - __setUpWaveEnergyConverterBuildOption, 124
 - __setUpWindTurbineBuildOption, 125
 - ~HexTile, 98
 - assess, 125
 - assets_manager_ptr, 132
 - build_menu_backing, 132
 - build_menu_backing_text, 132
 - build_menu_open, 133
 - build_menu_options_text_vec, 133
 - build_menu_options_vec, 133
 - credits, 133

- decorateTile, 126
- decoration_cleared, 133
- draw, 127
- draw_explosion, 133
- event_ptr, 134
- explosion_frame, 134
- explosion_sprite_reel, 134
- frame, 134
- game_phase, 134
- has_improvement, 134
- HexTile, 97
- is_selected, 135
- magnifying_glass_sprite, 135
- major_radius, 135
- message_hub_ptr, 135
- minor_radius, 135
- node_sprite, 135
- position_x, 136
- position_y, 136
- processEvent, 128
- processMessage, 129
- render_window_ptr, 136
- resource_assessed, 136
- resource_assessment, 136
- resource_chip_sprite, 136
- resource_text, 137
- scrap_improvement_frame, 137
- select_outline_sprite, 137
- setTileResource, 129, 130
- setTileType, 130, 131
- show_node, 137
- show_resource, 137
- tile_decoration_sprite, 137
- tile_improvement_ptr, 138
- tile_resource, 138
- tile_sprite, 138
- tile_type, 138
- toggleResourceOverlay, 132
- HexTile.h
 - ABOVE_AVERAGE, 241
 - AVERAGE, 241
 - BELOW_AVERAGE, 241
 - FOREST, 241
 - GOOD, 241
 - LAKE, 241
 - MOUNTAINS, 241
 - N_TILE_RESOURCES, 241
 - N_TILE_TYPES, 241
 - NONE_TYPE, 241
 - OCEAN, 241
 - PLAINS, 241
 - POOR, 241
 - TileResource, 240
 - TileType, 241
- int_payload
 - Message, 139
- is_running
 - TileImprovement, 188
- is_selected
 - HexTile, 135
 - TileImprovement, 188
- isEmpty
 - MessageHub, 143
- just_built
 - TileImprovement, 188
- just_upgraded
 - TileImprovement, 188
- LAKE
 - HexTile.h, 241
- LAKE_BLUE
 - constants.h, 220
- loadAssets
 - main.cpp, 258
- loadFont
 - AssetsManager, 14
- loadSound
 - AssetsManager, 14
- loadTexture
 - AssetsManager, 15
- loadTrack
 - AssetsManager, 16
- LOSS_CREDITS
 - Game.h, 238
- LOSS_DEMAND
 - Game.h, 238
- LOSS_EMISSIONS
 - Game.h, 238
- magnifying_glass_sprite
 - HexTile, 135
- main
 - main.cpp, 261
- main.cpp
 - constructRenderWindow, 258
 - loadAssets, 258
 - main, 261
- major_radius
 - HexTile, 135
- max_production_MWh
 - DieselGenerator, 45
- MAX_STORAGE_LEVELS
 - constants.h, 225
- MAX_UPGRADE_LEVELS
 - constants.h, 225
- MENU
 - ContextMenu.h, 214
- menu_frame
 - ContextMenu, 35
- MENU_FRAME_GREY
 - constants.h, 220
- Message, 138
 - bool_payload, 139
 - channel, 139
 - double_payload, 139
 - int_payload, 139

- string_payload, 140
- subject, 140
- message_hub
 - Game, 68
- message_hub_ptr
 - ContextMenu, 35
 - HexMap, 91
 - HexTile, 135
 - TileImprovement, 188
- message_map
 - MessageHub, 146
- MessageHub, 140
 - ~MessageHub, 141
 - addChannel, 141
 - clear, 142
 - clearMessages, 142
 - hasTraffic, 142
 - isEmpty, 143
 - message_map, 146
 - MessageHub, 141
 - popMessage, 143
 - receiveMessage, 144
 - removeChannel, 145
 - sendMessage, 145
- minor_radius
 - HexTile, 135
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, 220
- MONOCHROME_TEXT_AMBER
 - constants.h, 220
- MONOCHROME_TEXT_GREEN
 - constants.h, 221
- MONOCHROME_TEXT_RED
 - constants.h, 221
- month
 - Game, 68
- MOUNTAINS
 - HexTile.h, 241
- MOUNTAINS_GREY
 - constants.h, 221
- N_CONSOLE_STATES
 - ContextMenu.h, 214
- N_GAME_PHASES
 - Game.h, 238
- n_layers
 - HexMap, 91
- N_TILE_IMPROVEMENT_TYPES
 - TileImprovement.h, 246
- N_TILE_RESOURCES
 - HexTile.h, 241
- N_TILE_TYPES
 - HexTile.h, 241
- n_tiles
 - HexMap, 91
- nextTrack
 - AssetsManager, 16
- NO_TILE_SELECTED_CHANNEL
 - constants.h, 225
- node_sprite
 - HexTile, 135
- NONE_STATE
 - ContextMenu.h, 214
- NONE_TYPE
 - HexTile.h, 241
- OCEAN
 - HexTile.h, 241
- OCEAN_BLUE
 - constants.h, 221
- pauseTrack
 - AssetsManager, 17
- PLAINS
 - HexTile.h, 241
- PLAINS_YELLOW
 - constants.h, 221
- playTrack
 - AssetsManager, 17
- POOR
 - HexTile.h, 241
- popMessage
 - MessageHub, 143
- population
 - Game, 69
- position_x
 - ContextMenu, 36
 - HexMap, 91
 - HexTile, 136
 - TileImprovement, 188
- position_y
 - ContextMenu, 36
 - HexMap, 92
 - HexTile, 136
 - TileImprovement, 189
- previousTrack
 - AssetsManager, 17
- printGold
 - testing_utils.cpp, 252
 - testing_utils.h, 232
- printGreen
 - testing_utils.cpp, 252
 - testing_utils.h, 232
- printRed
 - testing_utils.cpp, 252
 - testing_utils.h, 233
- processEvent
 - ContextMenu, 32
 - DieselGenerator, 44
 - EnergyStorageSystem, 53
 - HexMap, 88
 - HexTile, 128
 - Settlement, 152
 - SolarPV, 162
 - TidalTurbine, 172
 - TileImprovement, 185
 - WaveEnergyConverter, 201
 - WindTurbine, 211

- processMessage
 - ContextMenu, 32
 - DieselGenerator, 44
 - EnergyStorageSystem, 53
 - HexMap, 88
 - HexTile, 129
 - Settlement, 152
 - SolarPV, 163
 - TidalTurbine, 172
 - TileImprovement, 186
 - WaveEnergyConverter, 201
 - WindTurbine, 211
- production_menu_backing
 - TileImprovement, 189
- production_menu_backing_text
 - TileImprovement, 189
- production_menu_open
 - TileImprovement, 189
- production_MWh
 - DieselGenerator, 45
 - SolarPV, 163
 - TidalTurbine, 173
 - WaveEnergyConverter, 202
 - WindTurbine, 212
- quit_game
 - Game, 69
- READY
 - ContextMenu.h, 214
- receiveMessage
 - MessageHub, 144
- removeChannel
 - MessageHub, 145
- render_window_ptr
 - ContextMenu, 36
 - Game, 69
 - HexMap, 92
 - HexTile, 136
 - TileImprovement, 189
- reroll
 - HexMap, 89
- resource_assessed
 - HexTile, 136
- resource_assessment
 - HexTile, 136
- RESOURCE_ASSESSMENT_COST
 - constants.h, 225
- RESOURCE_CHIP_GREY
 - constants.h, 222
- resource_chip_sprite
 - HexTile, 136
- resource_text
 - HexTile, 137
- run
 - Game, 66
- SCRAP_COST
 - constants.h, 225
- scrap_improvement_frame
 - HexTile, 137
- SECONDS_PER_FRAME
 - constants.h, 225
- SECONDS_PER_MONTH
 - constants.h, 226
- SECONDS_PER_YEAR
 - constants.h, 226
- select_outline_sprite
 - HexTile, 137
- sendMessage
 - MessageHub, 145
- setIsSelected
 - EnergyStorageSystem, 54
 - Settlement, 152
 - TileImprovement, 186
- setTileResource
 - HexTile, 129, 130
- setTileType
 - HexTile, 130, 131
- SETTLEMENT
 - TileImprovement.h, 246
- Settlement, 146
 - __handleKeyPressEvents, 149
 - __handleMouseButtonEvents, 149
 - __setUpTileImprovementSpriteStatic, 150
 - ~Settlement, 149
 - draw, 150
 - getTileOptionsSubstring, 151
 - processEvent, 152
 - processMessage, 152
 - setIsSelected, 152
 - Settlement, 148
 - smoke_da, 153
 - smoke_dx, 153
 - smoke_dy, 153
 - smoke_prob, 153
 - smoke_sprite_list, 154
- show_frame_clock_overlay
 - Game, 69
- show_node
 - HexTile, 137
- show_resource
 - HexMap, 92
 - HexTile, 137
- smoke_da
 - DieselGenerator, 45
 - Settlement, 153
- smoke_dx
 - DieselGenerator, 45
 - Settlement, 153
- smoke_dy
 - DieselGenerator, 46
 - Settlement, 153
- smoke_prob
 - DieselGenerator, 46
 - Settlement, 153
- smoke_sprite_list

- DieselGenerator, 46
- Settlement, 154
- SOLAR_PV
 - TileImprovement.h, 246
- SOLAR_PV_BUILD_COST
 - constants.h, 226
- SOLAR_PV_WATER_BUILD_MULTIPLIER
 - constants.h, 226
- SolarPV, 154
 - __drawUpgradeOptions, 157
 - __handleKeyPressEvents, 159
 - __handleMouseButtonEvents, 159
 - __setUpTileImprovementSpriteStatic, 160
 - __upgradePowerCapacity, 160
 - ~SolarPV, 157
 - capacity_kW, 163
 - dispatchable_MWh, 163
 - draw, 161
 - getTileOptionsSubstring, 162
 - processEvent, 162
 - processMessage, 163
 - production_MWh, 163
 - SolarPV, 156
- sound_map
 - AssetsManager, 18
- soundbuffer_map
 - AssetsManager, 18
- source/ContextMenu.cpp, 248
- source/DieselGenerator.cpp, 249
- source/EnergyStorageSystem.cpp, 249
- source/ESC_core/AssetsManager.cpp, 249
- source/ESC_core/MessageHub.cpp, 250
- source/ESC_core/testing_utils.cpp, 250
- source/Game.cpp, 256
- source/HexMap.cpp, 257
- source/HexTile.cpp, 257
- source/main.cpp, 258
- source/Settlement.cpp, 262
- source/SolarPV.cpp, 262
- source/TidalTurbine.cpp, 263
- source/TileImprovement.cpp, 263
- source/WaveEnergyConverter.cpp, 264
- source/WindTurbine.cpp, 264
- STARTING_CREDITS
 - constants.h, 226
- STARTING_POPULATION
 - constants.h, 226
- stopTrack
 - AssetsManager, 17
- storage_level
 - TileImprovement, 189
- storage_upgrade_sprite
 - TileImprovement, 190
- storage_upgrade_sprite_vec
 - TileImprovement, 190
- string_payload
 - Message, 140
- subject
 - Message, 140
- SYSTEM_MANAGEMENT
 - Game.h, 238
- testFloatEquals
 - testing_utils.cpp, 253
 - testing_utils.h, 233
- testGreaterThan
 - testing_utils.cpp, 253
 - testing_utils.h, 234
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 254
 - testing_utils.h, 234
- testing_utils.cpp
 - expectedErrorNotDetected, 251
 - printGold, 252
 - printGreen, 252
 - printRed, 252
 - testFloatEquals, 253
 - testGreaterThan, 253
 - testGreaterThanOrEqualTo, 254
 - testLessThan, 255
 - testLessThanOrEqualTo, 255
 - testTruth, 256
- testing_utils.h
 - expectedErrorNotDetected, 231
 - printGold, 232
 - printGreen, 232
 - printRed, 233
 - testFloatEquals, 233
 - testGreaterThan, 234
 - testGreaterThanOrEqualTo, 234
 - testLessThan, 235
 - testLessThanOrEqualTo, 236
 - testTruth, 236
- testLessThan
 - testing_utils.cpp, 255
 - testing_utils.h, 235
- testLessThanOrEqualTo
 - testing_utils.cpp, 255
 - testing_utils.h, 236
- testTruth
 - testing_utils.cpp, 256
 - testing_utils.h, 236
- texture_map
 - AssetsManager, 18
- TIDAL_TURBINE
 - TileImprovement.h, 246
- TIDAL_TURBINE_BUILD_COST
 - constants.h, 227
- TidalTurbine, 164
 - __drawUpgradeOptions, 167
 - __handleKeyPressEvents, 168
 - __handleMouseButtonEvents, 169
 - __setUpTileImprovementSpriteAnimated, 169
 - __upgradePowerCapacity, 170
 - ~TidalTurbine, 166
 - capacity_kW, 173
 - dispatchable_MWh, 173

- draw, 170
- getTileOptionsSubstring, 172
- processEvent, 172
- processMessage, 172
- production_MWh, 173
- TidalTurbine, 165
- TILE
 - ContextMenu.h, 214
- tile_decoration_sprite
 - HexTile, 137
- tile_improvement_ptr
 - HexTile, 138
- tile_improvement_sprite_animated
 - TileImprovement, 190
- tile_improvement_sprite_static
 - TileImprovement, 190
- tile_improvement_string
 - TileImprovement, 190
- tile_improvement_type
 - TileImprovement, 190
- tile_position_x_vec
 - HexMap, 92
- tile_position_y_vec
 - HexMap, 92
- tile_resource
 - HexTile, 138
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES
 - constants.h, 227
- tile_selected
 - HexMap, 92
- TILE_SELECTED_CHANNEL
 - constants.h, 227
- tile_sprite
 - HexTile, 138
- TILE_STATE_CHANNEL
 - constants.h, 227
- tile_type
 - HexTile, 138
- TILE_TYPE_CUMULATIVE_PROBABILITIES
 - constants.h, 227
- TileImprovement, 174
 - __closeProductionMenu, 178
 - __closeUpgradeMenu, 178
 - __handleKeyPressEvents, 179
 - __handleMouseButtonEvents, 179
 - __openProductionMenu, 180
 - __openUpgradeMenu, 180
 - __sendCreditsSpentMessage, 180
 - __sendGameStateRequest, 181
 - __sendInsufficientCreditsMessage, 181
 - __sendTileStateRequest, 181
 - __setUpProductionMenu, 182
 - __setUpUpgradeMenu, 182
 - __upgradeStorageCapacity, 183
 - ~TileImprovement, 178
 - assets_manager_ptr, 187
 - credits, 187
 - draw, 184
 - event_ptr, 187
 - frame, 187
 - game_phase, 187
 - getTileOptionsSubstring, 185
 - health, 187
 - is_running, 188
 - is_selected, 188
 - just_built, 188
 - just_upgraded, 188
 - message_hub_ptr, 188
 - position_x, 188
 - position_y, 189
 - processEvent, 185
 - processMessage, 186
 - production_menu_backing, 189
 - production_menu_backing_text, 189
 - production_menu_open, 189
 - render_window_ptr, 189
 - setIsSelected, 186
 - storage_level, 189
 - storage_upgrade_sprite, 190
 - storage_upgrade_sprite_vec, 190
 - tile_improvement_sprite_animated, 190
 - tile_improvement_sprite_static, 190
 - tile_improvement_string, 190
 - tile_improvement_type, 190
 - TileImprovement, 177
 - upgrade_arrow_sprite, 191
 - upgrade_frame, 191
 - upgrade_level, 191
 - upgrade_menu_backing, 191
 - upgrade_menu_backing_text, 191
 - upgrade_menu_open, 191
 - upgrade_plus_sprite, 192
- TileImprovement.h
 - DIESEL_GENERATOR, 246
 - ENERGY_STORAGE_SYSTEM, 246
 - N_TILE_IMPROVEMENT_TYPES, 246
 - SETTLEMENT, 246
 - SOLAR_PV, 246
 - TIDAL_TURBINE, 246
 - TileImprovementType, 245
 - WAVE_ENERGY_CONVERTER, 246
 - WIND_TURBINE, 246
- TileImprovementType
 - TileImprovement.h, 245
- TileResource
 - HexTile.h, 240
- TileType
 - HexTile.h, 241
- time_since_start_s
 - Game, 69
- toggleResourceOverlay
 - HexMap, 89
 - HexTile, 132
- track_map
 - AssetsManager, 19
- turn

- Game, 69
- upgrade_arrow_sprite
 - TileImprovement, 191
- upgrade_frame
 - TileImprovement, 191
- upgrade_level
 - TileImprovement, 191
- upgrade_menu_backing
 - TileImprovement, 191
- upgrade_menu_backing_text
 - TileImprovement, 191
- upgrade_menu_open
 - TileImprovement, 191
- upgrade_plus_sprite
 - TileImprovement, 192
- VICTORY
 - Game.h, 238
- visual_screen
 - ContextMenu, 36
- visual_screen_frame_bottom
 - ContextMenu, 36
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, 222
- visual_screen_frame_left
 - ContextMenu, 36
- visual_screen_frame_right
 - ContextMenu, 37
- visual_screen_frame_top
 - ContextMenu, 37
- WAVE_ENERGY_CONVERTER
 - TileImprovement.h, 246
- WAVE_ENERGY_CONVERTER_BUILD_COST
 - constants.h, 228
- WaveEnergyConverter, 192
 - __drawUpgradeOptions, 195
 - __handleKeyPressEvents, 197
 - __handleMouseButtonEvents, 197
 - __setUpTileImprovementSpriteAnimated, 198
 - __upgradePowerCapacity, 198
 - ~WaveEnergyConverter, 195
 - capacity_kW, 202
 - dispatchable_MWh, 202
 - draw, 199
 - getTileOptionsSubstring, 200
 - processEvent, 201
 - processMessage, 201
 - production_MWh, 202
 - WaveEnergyConverter, 194
- WIND_TURBINE
 - TileImprovement.h, 246
- WIND_TURBINE_BUILD_COST
 - constants.h, 228
- WIND_TURBINE_WATER_BUILD_MULTIPLIER
 - constants.h, 228
- WindTurbine, 202
 - __drawUpgradeOptions, 205
 - __handleKeyPressEvents, 207
 - __handleMouseButtonEvents, 207
 - __setUpTileImprovementSpriteAnimated, 208
 - __upgradePowerCapacity, 208
 - ~WindTurbine, 205
 - capacity_kW, 212
 - dispatchable_MWh, 212
 - draw, 209
 - getTileOptionsSubstring, 210
 - processEvent, 211
 - processMessage, 211
 - production_MWh, 212
 - WindTurbine, 204
- year
 - Game, 70