

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()	40
4.3.2.2 ~DieselGenerator()	41
4.3.3 Member Function Documentation	41
4.3.3.1 __breakdown()	41
4.3.3.2 __computeProductionCosts()	41
4.3.3.3 __drawProductionMenu()	42
4.3.3.4 __handleKeyPressEvents()	42
4.3.3.5 __handleMouseButtonEvents()	43
4.3.3.6 __sendImprovementStateMessage()	44
4.3.3.7 __setUpTileImprovementSpriteAnimated()	44
4.3.3.8 __upgrade()	45
4.3.3.9 advanceTurn()	45
4.3.3.10 draw()	46
4.3.3.11 getTileOptionsSubstring()	48
4.3.3.12 processEvent()	49
4.3.3.13 processMessage()	49
4.3.3.14 setIsSelected()	49
4.3.4 Member Data Documentation	50
4.3.4.1 capacity_kW	50
4.3.4.2 emissions_tonnes_CO2e	50
4.3.4.3 fuel_cost	50
4.3.4.4 max_production_MWh	50
4.3.4.5 production_MWh	51
4.3.4.6 smoke_da	51
4.3.4.7 smoke_dx	51
4.3.4.8 smoke_dy	51
4.3.4.9 smoke_prob	51
4.3.4.10 smoke_sprite_list	51
4.4 EnergyStorageSystem Class Reference	52
4.4.1 Detailed Description	53
4.4.2 Constructor & Destructor Documentation	53
4.4.2.1 EnergyStorageSystem()	53
4.4.2.2 ~EnergyStorageSystem()	54
4.4.3 Member Function Documentation	54
4.4.3.1 __handleKeyPressEvents()	55
4.4.3.2 __handleMouseButtonEvents()	55
4.4.3.3 __setUpProductionMenu()	56

4.4.3.4 __setUpTileImprovementSpriteStatic()	56
4.4.3.5 __upgrade()	56
4.4.3.6 draw()	57
4.4.3.7 getTileOptionsSubstring()	57
4.4.3.8 processEvent()	58
4.4.3.9 processMessage()	59
4.4.3.10 setIsSelected()	59
4.4.4 Member Data Documentation	59
4.4.4.1 capacity_MWh	59
4.4.4.2 charge_MWh	60
4.5 Game Class Reference	60
4.5.1 Detailed Description	62
4.5.2 Constructor & Destructor Documentation	63
4.5.2.1 Game()	63
4.5.2.2 ~Game()	64
4.5.3 Member Function Documentation	64
4.5.3.1 __advanceTurn()	64
4.5.3.2 __checkTerminatingConditions()	64
4.5.3.3 __computeCurrentDemand()	65
4.5.3.4 __draw()	65
4.5.3.5 __drawFrameClockOverlay()	65
4.5.3.6 __drawHUD()	66
4.5.3.7 __handleImprovementStateMessage()	67
4.5.3.8 __handleKeyPressEvents()	68
4.5.3.9 __handleMouseButtonEvents()	68
4.5.3.10 __insufficientCreditsAlarm()	69
4.5.3.11 __processEvent()	70
4.5.3.12 __processMessage()	71
4.5.3.13 __sendCreditsEarnedMessage()	72
4.5.3.14 __sendGameStateMessage()	72
4.5.3.15 __sendTurnAdvanceMessage()	73
4.5.3.16 __toggleFrameClockOverlay()	74
4.5.3.17 run()	74
4.5.4 Member Data Documentation	75
4.5.4.1 assets_manager_ptr	75
4.5.4.2 check_terminating_conditions	75
4.5.4.3 clock	75
4.5.4.4 context_menu_ptr	76
4.5.4.5 credits	76
4.5.4.6 cumulative_emissions_tonnes	76
4.5.4.7 demand_MWh	76
4.5.4.8 demand_remaining_MWh	76

4.5.4.9 demand_vec_MWh	76
4.5.4.10 event	77
4.5.4.11 frame	77
4.5.4.12 game_loop_broken	77
4.5.4.13 game_phase	77
4.5.4.14 hex_map_ptr	77
4.5.4.15 message_deadlock	77
4.5.4.16 message_deadlock_frame	78
4.5.4.17 message_hub	78
4.5.4.18 month	78
4.5.4.19 population	78
4.5.4.20 quit_game	78
4.5.4.21 render_window_ptr	78
4.5.4.22 show_frame_clock_overlay	79
4.5.4.23 time_since_start_s	79
4.5.4.24 turn	79
4.5.4.25 year	79
4.6 HexMap Class Reference	79
4.6.1 Detailed Description	82
4.6.2 Constructor & Destructor Documentation	82
4.6.2.1 HexMap()	82
4.6.2.2 ~HexMap()	83
4.6.3 Member Function Documentation	83
4.6.3.1 __assembleHexMap()	83
4.6.3.2 __assessNeighbours()	83
4.6.3.3 __buildDrawOrderVector()	84
4.6.3.4 __enforceOceanContinuity()	85
4.6.3.5 __getMajorityTileType()	85
4.6.3.6 __getNeighboursVector()	86
4.6.3.7 __getNoise()	87
4.6.3.8 __getSelectedTile()	88
4.6.3.9 __getValidMapIndexPositions()	89
4.6.3.10 __handleKeyPressEvents()	90
4.6.3.11 __handleMouseButtonEvents()	90
4.6.3.12 __isLakeTouchingOcean()	91
4.6.3.13 __layTiles()	91
4.6.3.14 __procedurallyGenerateTileResources()	93
4.6.3.15 __procedurallyGenerateTileTypes()	94
4.6.3.16 __sendNoTileSelectedMessage()	95
4.6.3.17 __setUpGlassScreen()	95
4.6.3.18 __smoothTileTypes()	95
4.6.3.19 assess()	96

4.6.3.20 clear()	96
4.6.3.21 draw()	96
4.6.3.22 processEvent()	97
4.6.3.23 processMessage()	98
4.6.3.24 reroll()	98
4.6.3.25 toggleResourceOverlay()	98
4.6.4 Member Data Documentation	99
4.6.4.1 assets_manager_ptr	99
4.6.4.2 border_tiles_vec	99
4.6.4.3 event_ptr	99
4.6.4.4 frame	99
4.6.4.5 glass_screen	100
4.6.4.6 hex_draw_order_vec	100
4.6.4.7 hex_map	100
4.6.4.8 message_hub_ptr	100
4.6.4.9 n_layers	100
4.6.4.10 n_tiles	100
4.6.4.11 position_x	101
4.6.4.12 position_y	101
4.6.4.13 render_window_ptr	101
4.6.4.14 show_resource	101
4.6.4.15 tile_position_x_vec	101
4.6.4.16 tile_position_y_vec	101
4.6.4.17 tile_selected	102
4.7 HexTile Class Reference	102
4.7.1 Detailed Description	106
4.7.2 Constructor & Destructor Documentation	106
4.7.2.1 HexTile()	106
4.7.2.2 ~HexTile()	107
4.7.3 Member Function Documentation	108
4.7.3.1 __buildDieselGenerator()	108
4.7.3.2 __buildEnergyStorage()	108
4.7.3.3 __buildSettlement()	109
4.7.3.4 __buildSolarPV()	109
4.7.3.5 __buildTidalTurbine()	110
4.7.3.6 __buildWaveEnergyConverter()	111
4.7.3.7 __buildWindTurbine()	111
4.7.3.8 __clearDecoration()	112
4.7.3.9 __closeBuildMenu()	112
4.7.3.10 __getTileCoordsSubstring()	113
4.7.3.11 __getTileImprovementSubstring()	113
4.7.3.12 __getTileOptionsSubstring()	113

4.7.3.13	__getTileResourceSubstring()	115
4.7.3.14	__getTileTypeSubstring()	116
4.7.3.15	__handleKeyPressEvents()	116
4.7.3.16	__handleKeyReleaseEvents()	120
4.7.3.17	__handleMouseButtonEvents()	121
4.7.3.18	__isClicked()	122
4.7.3.19	__openBuildMenu()	122
4.7.3.20	__scrapImprovement()	122
4.7.3.21	__sendAssessNeighboursMessage()	123
4.7.3.22	__sendCreditsSpentMessage()	124
4.7.3.23	__sendGameStateRequest()	124
4.7.3.24	__sendInsufficientCreditsMessage()	124
4.7.3.25	__sendTileSelectedMessage()	125
4.7.3.26	__sendTileStateMessage()	125
4.7.3.27	__sendUpdateGamePhaseMessage()	125
4.7.3.28	__setIsSelected()	126
4.7.3.29	__setResourceText()	126
4.7.3.30	__setUpBuildMenu()	127
4.7.3.31	__setUpBuildOption()	128
4.7.3.32	__setUpDieselGeneratorBuildOption()	129
4.7.3.33	__setUpEnergyStorageSystemBuildOption()	130
4.7.3.34	__setUpMagnifyingGlassSprite()	130
4.7.3.35	__setUpNodeSprite()	131
4.7.3.36	__setUpResourceChipSprite()	131
4.7.3.37	__setUpSelectOutlineSprite()	131
4.7.3.38	__setUpSolarPVBuildOption()	132
4.7.3.39	__setUpTidalTurbineBuildOption()	132
4.7.3.40	__setUpTileExplosionReel()	133
4.7.3.41	__setUpTileSprite()	133
4.7.3.42	__setUpWaveEnergyConverterBuildOption()	133
4.7.3.43	__setUpWindTurbineBuildOption()	134
4.7.3.44	assess()	135
4.7.3.45	decorateTile()	135
4.7.3.46	draw()	136
4.7.3.47	processEvent()	137
4.7.3.48	processMessage()	138
4.7.3.49	setTileResource() [1/2]	139
4.7.3.50	setTileResource() [2/2]	139
4.7.3.51	setTileType() [1/2]	140
4.7.3.52	setTileType() [2/2]	140
4.7.3.53	toggleResourceOverlay()	141
4.7.4	Member Data Documentation	141

4.7.4.1 assets_manager_ptr	141
4.7.4.2 build_menu_backing	142
4.7.4.3 build_menu_backing_text	142
4.7.4.4 build_menu_open	142
4.7.4.5 build_menu_options_text_vec	142
4.7.4.6 build_menu_options_vec	142
4.7.4.7 credits	142
4.7.4.8 decoration_cleared	143
4.7.4.9 draw_explosion	143
4.7.4.10 event_ptr	143
4.7.4.11 explosion_frame	143
4.7.4.12 explosion_sprite_reel	143
4.7.4.13 frame	143
4.7.4.14 game_phase	144
4.7.4.15 has_improvement	144
4.7.4.16 is_selected	144
4.7.4.17 magnifying_glass_sprite	144
4.7.4.18 major_radius	144
4.7.4.19 message_hub_ptr	144
4.7.4.20 minor_radius	145
4.7.4.21 node_sprite	145
4.7.4.22 position_x	145
4.7.4.23 position_y	145
4.7.4.24 render_window_ptr	145
4.7.4.25 resource_assessed	145
4.7.4.26 resource_assessment	146
4.7.4.27 resource_chip_sprite	146
4.7.4.28 resource_text	146
4.7.4.29 scrap_improvement_frame	146
4.7.4.30 select_outline_sprite	146
4.7.4.31 show_node	146
4.7.4.32 show_resource	147
4.7.4.33 tile_decoration_sprite	147
4.7.4.34 tile_improvement_ptr	147
4.7.4.35 tile_resource	147
4.7.4.36 tile_sprite	147
4.7.4.37 tile_type	147
4.8 Message Struct Reference	148
4.8.1 Detailed Description	148
4.8.2 Member Data Documentation	148
4.8.2.1 bool_payload	148
4.8.2.2 channel	148

4.8.2.3 double_payload	149
4.8.2.4 int_payload	149
4.8.2.5 string_payload	149
4.8.2.6 subject	149
4.8.2.7 vector_payload	149
4.9 MessageHub Class Reference	149
4.9.1 Detailed Description	150
4.9.2 Constructor & Destructor Documentation	150
4.9.2.1 MessageHub()	151
4.9.2.2 ~MessageHub()	151
4.9.3 Member Function Documentation	151
4.9.3.1 addChannel()	151
4.9.3.2 clear()	152
4.9.3.3 clearMessages()	152
4.9.3.4 hasTraffic()	152
4.9.3.5 isEmpty()	153
4.9.3.6 popMessage()	154
4.9.3.7 printState()	155
4.9.3.8 receiveMessage()	155
4.9.3.9 removeChannel()	156
4.9.3.10 sendMessage()	157
4.9.4 Member Data Documentation	157
4.9.4.1 message_map	157
4.10 Settlement Class Reference	157
4.10.1 Detailed Description	159
4.10.2 Constructor & Destructor Documentation	159
4.10.2.1 Settlement()	159
4.10.2.2 ~Settlement()	160
4.10.3 Member Function Documentation	160
4.10.3.1 __handleKeyPressEvents()	161
4.10.3.2 __handleMouseButtonEvents()	161
4.10.3.3 __setUpCoinSprite()	162
4.10.3.4 __setUpTileImprovementSpriteStatic()	162
4.10.3.5 draw()	162
4.10.3.6 getTileOptionsSubstring()	163
4.10.3.7 processEvent()	164
4.10.3.8 processMessage()	164
4.10.3.9 setIsSelected()	165
4.10.4 Member Data Documentation	165
4.10.4.1 coin_sprite	165
4.10.4.2 draw_coin	165
4.10.4.3 smoke_da	165

4.10.4.4 smoke_dx	166
4.10.4.5 smoke_dy	166
4.10.4.6 smoke_prob	166
4.10.4.7 smoke_sprite_list	166
4.11 SolarPV Class Reference	167
4.11.1 Detailed Description	169
4.11.2 Constructor & Destructor Documentation	169
4.11.2.1 SolarPV()	169
4.11.2.2 ~SolarPV()	170
4.11.3 Member Function Documentation	170
4.11.3.1 __breakdown()	170
4.11.3.2 __computeCapacityFactors()	171
4.11.3.3 __computeDispatch()	171
4.11.3.4 __computeProduction()	172
4.11.3.5 __computeProductionCosts()	173
4.11.3.6 __drawProductionMenu()	173
4.11.3.7 __drawUpgradeOptions()	174
4.11.3.8 __handleKeyPressEvents()	175
4.11.3.9 __handleMouseButtonEvents()	176
4.11.3.10 __sendImprovementStateMessage()	176
4.11.3.11 __setUpTileImprovementSpriteStatic()	177
4.11.3.12 __upgradePowerCapacity()	177
4.11.3.13 advanceTurn()	178
4.11.3.14 draw()	178
4.11.3.15 getTileOptionsSubstring()	180
4.11.3.16 processEvent()	180
4.11.3.17 processMessage()	181
4.11.3.18 setIsSelected()	181
4.11.3.19 update()	181
4.11.4 Member Data Documentation	182
4.11.4.1 capacity_factor_vec	182
4.11.4.2 capacity_kW	182
4.11.4.3 dispatch_MWh	182
4.11.4.4 dispatch_vec_MWh	182
4.11.4.5 dispatchable_MWh	183
4.11.4.6 max_daily_production_MWh	183
4.11.4.7 production_MWh	183
4.11.4.8 production_vec_MWh	183
4.12 TidalTurbine Class Reference	184
4.12.1 Detailed Description	186
4.12.2 Constructor & Destructor Documentation	186
4.12.2.1 TidalTurbine()	186

4.12.2.2 ~TidalTurbine()	187
4.12.3 Member Function Documentation	187
4.12.3.1 __breakdown()	188
4.12.3.2 __computeCapacityFactors()	188
4.12.3.3 __computeDispatch()	188
4.12.3.4 __computeProduction()	189
4.12.3.5 __computeProductionCosts()	189
4.12.3.6 __drawProductionMenu()	190
4.12.3.7 __drawUpgradeOptions()	190
4.12.3.8 __handleKeyPressEvents()	192
4.12.3.9 __handleMouseButtonEvents()	193
4.12.3.10 __sendImprovementStateMessage()	193
4.12.3.11 __setUpTileImprovementSpriteAnimated()	194
4.12.3.12 __upgradePowerCapacity()	194
4.12.3.13 advanceTurn()	195
4.12.3.14 draw()	195
4.12.3.15 getTileOptionsSubstring()	197
4.12.3.16 processEvent()	198
4.12.3.17 processMessage()	198
4.12.3.18 setIsSelected()	198
4.12.3.19 update()	199
4.12.4 Member Data Documentation	199
4.12.4.1 bobbing_y	199
4.12.4.2 capacity_factor_vec	199
4.12.4.3 capacity_kW	199
4.12.4.4 dispatch_MWh	200
4.12.4.5 dispatch_vec_MWh	200
4.12.4.6 dispatchable_MWh	200
4.12.4.7 max_daily_production_MWh	200
4.12.4.8 production_MWh	200
4.12.4.9 production_vec_MWh	200
4.12.4.10 rotor_drotation	201
4.13 TileImprovement Class Reference	201
4.13.1 Detailed Description	205
4.13.2 Constructor & Destructor Documentation	205
4.13.2.1 TileImprovement()	205
4.13.2.2 ~TileImprovement()	206
4.13.3 Member Function Documentation	207
4.13.3.1 __breakdown()	207
4.13.3.2 __closeProductionMenu()	207
4.13.3.3 __closeUpgradeMenu()	207
4.13.3.4 __drawDispatch()	208

4.13.3.5	__handleKeyPressEvents()	208
4.13.3.6	__handleMouseButtonEvents()	209
4.13.3.7	__openProductionMenu()	209
4.13.3.8	__openUpgradeMenu()	210
4.13.3.9	__repair()	210
4.13.3.10	__sendCreditsSpentMessage()	210
4.13.3.11	__sendGameStateRequest()	211
4.13.3.12	__sendInsufficientCreditsMessage()	211
4.13.3.13	__sendTileStateRequest()	211
4.13.3.14	__setUpDispatchIllustration()	212
4.13.3.15	__setUpProductionMenu()	212
4.13.3.16	__setUpUpgradeMenu()	213
4.13.3.17	__upgradeStorageCapacity()	213
4.13.3.18	advanceTurn()	214
4.13.3.19	draw()	214
4.13.3.20	getTileOptionsSubstring()	216
4.13.3.21	processEvent()	216
4.13.3.22	processMessage()	217
4.13.3.23	setIsSelected()	217
4.13.3.24	update()	217
4.13.4	Member Data Documentation	218
4.13.4.1	assets_manager_ptr	218
4.13.4.2	credits	218
4.13.4.3	demand_MWh	218
4.13.4.4	demand_vec_MWh	218
4.13.4.5	dispatch_backing	218
4.13.4.6	dispatch_text	219
4.13.4.7	event_ptr	219
4.13.4.8	frame	219
4.13.4.9	game_phase	219
4.13.4.10	health	219
4.13.4.11	is_broken	219
4.13.4.12	is_running	220
4.13.4.13	is_selected	220
4.13.4.14	just_built	220
4.13.4.15	just_upgraded	220
4.13.4.16	message_hub_ptr	220
4.13.4.17	month	220
4.13.4.18	operation_maintenance_cost	221
4.13.4.19	position_x	221
4.13.4.20	position_y	221
4.13.4.21	production_menu_backing	221

4.13.4.22 production_menu_backing_text	221
4.13.4.23 production_menu_open	221
4.13.4.24 render_window_ptr	222
4.13.4.25 storage_kWh	222
4.13.4.26 storage_level	222
4.13.4.27 storage_upgrade_sprite	222
4.13.4.28 storage_upgrade_sprite_vec	222
4.13.4.29 tile_improvement_sprite_animated	222
4.13.4.30 tile_improvement_sprite_static	223
4.13.4.31 tile_improvement_string	223
4.13.4.32 tile_improvement_type	223
4.13.4.33 tile_resource	223
4.13.4.34 tile_resource_scalar	223
4.13.4.35 upgrade_arrow_sprite	223
4.13.4.36 upgrade_frame	224
4.13.4.37 upgrade_level	224
4.13.4.38 upgrade_menu_backing	224
4.13.4.39 upgrade_menu_backing_text	224
4.13.4.40 upgrade_menu_open	224
4.13.4.41 upgrade_plus_sprite	224
4.14 WaveEnergyConverter Class Reference	225
4.14.1 Detailed Description	227
4.14.2 Constructor & Destructor Documentation	227
4.14.2.1 WaveEnergyConverter()	227
4.14.2.2 ~WaveEnergyConverter()	228
4.14.3 Member Function Documentation	228
4.14.3.1 __breakdown()	229
4.14.3.2 __computeCapacityFactors()	229
4.14.3.3 __computeDispatch()	229
4.14.3.4 __computeProduction()	230
4.14.3.5 __computeProductionCosts()	231
4.14.3.6 __drawProductionMenu()	231
4.14.3.7 __drawUpgradeOptions()	232
4.14.3.8 __handleKeyPressEvents()	233
4.14.3.9 __handleMouseButtonEvents()	234
4.14.3.10 __sendImprovementStateMessage()	235
4.14.3.11 __setUpTileImprovementSpriteAnimated()	235
4.14.3.12 __upgradePowerCapacity()	236
4.14.3.13 advanceTurn()	236
4.14.3.14 draw()	237
4.14.3.15 getTileOptionsSubstring()	238
4.14.3.16 processEvent()	239

4.14.3.17 processMessage()	240
4.14.3.18 setIsSelected()	240
4.14.3.19 update()	240
4.14.4 Member Data Documentation	241
4.14.4.1 bobbing_y	241
4.14.4.2 capacity_factor_vec	241
4.14.4.3 capacity_kW	241
4.14.4.4 dispatch_MWh	241
4.14.4.5 dispatch_vec_MWh	241
4.14.4.6 dispatchable_MWh	242
4.14.4.7 max_daily_production_MWh	242
4.14.4.8 production_MWh	242
4.14.4.9 production_vec_MWh	242
4.15 WindTurbine Class Reference	243
4.15.1 Detailed Description	245
4.15.2 Constructor & Destructor Documentation	245
4.15.2.1 WindTurbine()	245
4.15.2.2 ~WindTurbine()	246
4.15.3 Member Function Documentation	246
4.15.3.1 __breakdown()	247
4.15.3.2 __computeCapacityFactors()	247
4.15.3.3 __computeDispatch()	247
4.15.3.4 __computeProduction()	248
4.15.3.5 __computeProductionCosts()	249
4.15.3.6 __drawProductionMenu()	249
4.15.3.7 __drawUpgradeOptions()	250
4.15.3.8 __handleKeyPressEvents()	251
4.15.3.9 __handleMouseButtonEvents()	252
4.15.3.10 __sendImprovementStateMessage()	253
4.15.3.11 __setUpTileImprovementSpriteAnimated()	253
4.15.3.12 __upgradePowerCapacity()	254
4.15.3.13 advanceTurn()	254
4.15.3.14 draw()	255
4.15.3.15 getTileOptionsSubstring()	256
4.15.3.16 processEvent()	257
4.15.3.17 processMessage()	257
4.15.3.18 setIsSelected()	258
4.15.3.19 update()	258
4.15.4 Member Data Documentation	258
4.15.4.1 capacity_factor_vec	258
4.15.4.2 capacity_kW	259
4.15.4.3 dispatch_MWh	259

4.15.4.4 dispatch_vec_MWh	259
4.15.4.5 dispatchable_MWh	259
4.15.4.6 max_daily_production_MWh	259
4.15.4.7 production_MWh	259
4.15.4.8 production_vec_MWh	260
4.15.4.9 rotor_drotation	260
5 File Documentation	261
5.1 header/ContextMenu.h File Reference	261
5.1.1 Detailed Description	262
5.1.2 Enumeration Type Documentation	262
5.1.2.1 ConsoleState	262
5.2 header/DieselGenerator.h File Reference	262
5.2.1 Detailed Description	263
5.3 header/EnergyStorageSystem.h File Reference	263
5.3.1 Detailed Description	264
5.4 header/ESC_core/AssetsManager.h File Reference	264
5.4.1 Detailed Description	265
5.5 header/ESC_core/constants.h File Reference	265
5.5.1 Detailed Description	268
5.5.2 Function Documentation	269
5.5.2.1 FOREST_GREEN()	269
5.5.2.2 LAKE_BLUE()	269
5.5.2.3 MENU_FRAME_GREY()	269
5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()	269
5.5.2.5 MONOCHROME_TEXT_AMBER()	270
5.5.2.6 MONOCHROME_TEXT_GREEN()	270
5.5.2.7 MONOCHROME_TEXT_RED()	270
5.5.2.8 MOUNTAINS_GREY()	270
5.5.2.9 OCEAN_BLUE()	270
5.5.2.10 PLAINS_YELLOW()	271
5.5.2.11 RESOURCE_CHIP_GREY()	271
5.5.2.12 VISUAL_SCREEN_FRAME_GREY()	271
5.5.3 Variable Documentation	271
5.5.3.1 BUILD_SETTLEMENT_COST	271
5.5.3.2 CLEAR_FOREST_COST	271
5.5.3.3 CLEAR_MOUNTAINS_COST	272
5.5.3.4 CLEAR_PLAINS_COST	272
5.5.3.5 COST_PER_LITRE_DIESEL	272
5.5.3.6 CREDITS_PER_MWH_SERVED	272
5.5.3.7 DAILY_TIDAL_CAPACITY_FACTOR	272
5.5.3.8 DIESEL_GENERATOR_BUILD_COST	272

5.5.3.9 DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION	273
5.5.3.10 EMISSIONS_LIFETIME_LIMIT_TONNES	273
5.5.3.11 ENERGY_STORAGE_SYSTEM_BUILD_COST	273
5.5.3.12 FLOAT_TOLERANCE	273
5.5.3.13 FRAMES_PER_SECOND	273
5.5.3.14 GAME_CHANNEL	273
5.5.3.15 GAME_HEIGHT	274
5.5.3.16 GAME_STATE_CHANNEL	274
5.5.3.17 GAME_WIDTH	274
5.5.3.18 HEX_MAP_CHANNEL	274
5.5.3.19 KG_CO2E_PER_LITRE_DIESEL	274
5.5.3.20 LITRES_DIESEL_PER_MWH_PRODUCTION	274
5.5.3.21 MAX_STORAGE_LEVELS	275
5.5.3.22 MAX_UPGRADE_LEVELS	275
5.5.3.23 MAXIMUM_DAILY_DEMAND_PER_CAPITA	275
5.5.3.24 MEAN_DAILY_DEMAND_RATIOS	275
5.5.3.25 MEAN_DAILY_SOLAR_CAPACITY_FACTORS	275
5.5.3.26 MEAN_DAILY_WAVE_CAPACITY_FACTORS	276
5.5.3.27 MEAN_DAILY_WIND_CAPACITY_FACTORS	276
5.5.3.28 NO_TILE_SELECTED_CHANNEL	276
5.5.3.29 POPULATION_MONTHLY_GROWTH_RATE	276
5.5.3.30 RESOURCE_ASSESSMENT_COST	276
5.5.3.31 SCRAP_COST	277
5.5.3.32 SECONDS_PER_FRAME	277
5.5.3.33 SECONDS_PER_MONTH	277
5.5.3.34 SECONDS_PER_YEAR	277
5.5.3.35 SETTLEMENT_CHANNEL	277
5.5.3.36 SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION	277
5.5.3.37 SOLAR_PV_BUILD_COST	278
5.5.3.38 SOLAR_PV_WATER_BUILD_MULTIPLIER	278
5.5.3.39 STARTING_CREDITS	278
5.5.3.40 STARTING_POPULATION	278
5.5.3.41 STDEV_DAILY_DEMAND_RATIOS	278
5.5.3.42 STDEV_DAILY_SOLAR_CAPACITY_FACTORS	279
5.5.3.43 STDEV_DAILY_WAVE_CAPACITY_FACTORS	279
5.5.3.44 STDEV_DAILY_WIND_CAPACITY_FACTORS	279
5.5.3.45 TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION	279
5.5.3.46 TIDAL_TURBINE_BUILD_COST	280
5.5.3.47 TILE_RESOURCE_CUMULATIVE_PROBABILITIES	280
5.5.3.48 TILE_SELECTED_CHANNEL	280
5.5.3.49 TILE_STATE_CHANNEL	280
5.5.3.50 TILE_TYPE_CUMULATIVE_PROBABILITIES	280

5.5.3.51 WAVE_ENERGY_CONVERTER_BUILD_COST	281
5.5.3.52 WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION	281
5.5.3.53 WIND_OP_MAINT_COST_PER_MWH_PRODUCTION	281
5.5.3.54 WIND_TURBINE_BUILD_COST	281
5.5.3.55 WIND_TURBINE_WATER_BUILD_MULTIPLIER	281
5.6 header/ESC_core/doxygen_cite.h File Reference	281
5.6.1 Detailed Description	281
5.7 header/ESC_core/includes.h File Reference	282
5.7.1 Detailed Description	282
5.8 header/ESC_core/MessageHub.h File Reference	283
5.8.1 Detailed Description	283
5.9 header/ESC_core/testing_utils.h File Reference	283
5.9.1 Detailed Description	284
5.9.2 Function Documentation	284
5.9.2.1 expectedErrorNotDetected()	285
5.9.2.2 printGold()	285
5.9.2.3 printGreen()	285
5.9.2.4 printRed()	286
5.9.2.5 testFloatEquals()	286
5.9.2.6 testGreaterThan()	287
5.9.2.7 testGreaterThanOrEqualTo()	287
5.9.2.8 testLessThan()	288
5.9.2.9 testLessThanOrEqualTo()	289
5.9.2.10 testTruth()	289
5.10 header/Game.h File Reference	290
5.10.1 Enumeration Type Documentation	291
5.10.1.1 GamePhase	291
5.11 header/HexMap.h File Reference	291
5.11.1 Detailed Description	292
5.12 header/HexTile.h File Reference	292
5.12.1 Detailed Description	293
5.12.2 Enumeration Type Documentation	293
5.12.2.1 TileResource	293
5.12.2.2 TileType	294
5.13 header/Settlement.h File Reference	294
5.13.1 Detailed Description	295
5.14 header/SolarPV.h File Reference	295
5.14.1 Detailed Description	296
5.15 header/TidalTurbine.h File Reference	296
5.15.1 Detailed Description	297
5.16 header/TileImprovement.h File Reference	297
5.16.1 Detailed Description	298

5.16.2 Enumeration Type Documentation	298
5.16.2.1 TileImprovementType	298
5.17 header/WaveEnergyConverter.h File Reference	299
5.17.1 Detailed Description	300
5.18 header/WindTurbine.h File Reference	300
5.18.1 Detailed Description	301
5.19 source/ContextMenu.cpp File Reference	301
5.19.1 Detailed Description	301
5.20 source/DieselGenerator.cpp File Reference	301
5.20.1 Detailed Description	301
5.21 source/EnergyStorageSystem.cpp File Reference	302
5.21.1 Detailed Description	302
5.22 source/ESC_core/AssetsManager.cpp File Reference	302
5.22.1 Detailed Description	302
5.23 source/ESC_core/MessageHub.cpp File Reference	302
5.23.1 Detailed Description	303
5.24 source/ESC_core/testing_utils.cpp File Reference	303
5.24.1 Detailed Description	303
5.24.2 Function Documentation	304
5.24.2.1 expectedErrorNotDetected()	304
5.24.2.2 printGold()	304
5.24.2.3 printGreen()	304
5.24.2.4 printRed()	305
5.24.2.5 testFloatEquals()	305
5.24.2.6 testGreaterThan()	306
5.24.2.7 testGreaterThanOrEqualTo()	306
5.24.2.8 testLessThan()	307
5.24.2.9 testLessThanOrEqualTo()	308
5.24.2.10 testTruth()	308
5.25 source/Game.cpp File Reference	309
5.25.1 Detailed Description	309
5.26 source/HexMap.cpp File Reference	309
5.26.1 Detailed Description	310
5.27 source/HexTile.cpp File Reference	310
5.27.1 Detailed Description	310
5.28 source/main.cpp File Reference	310
5.28.1 Detailed Description	311
5.28.2 Function Documentation	311
5.28.2.1 constructRenderWindow()	311
5.28.2.2 loadAssets()	311
5.28.2.3 main()	314
5.29 source/Settlement.cpp File Reference	315

5.29.1 Detailed Description	315
5.30 source/SolarPV.cpp File Reference	315
5.30.1 Detailed Description	315
5.31 source/TidalTurbine.cpp File Reference	316
5.31.1 Detailed Description	316
5.32 source/TileImprovement.cpp File Reference	316
5.32.1 Detailed Description	316
5.33 source/WaveEnergyConverter.cpp File Reference	316
5.33.1 Detailed Description	317
5.34 source/WindTurbine.cpp File Reference	317
5.34.1 Detailed Description	317
Bibliography	319
Index	321

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

AssetsManager	7
ContextMenu	19
Game	60
HexMap	79
HexTile	102
Message	148
MessageHub	149
TileImprovement	201
DieselGenerator	37
EnergyStorageSystem	52
Settlement	157
SolarPV	167
TidalTurbine	184
WaveEnergyConverter	225
WindTurbine	243

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)	52
Game	A class which acts as the central class for the game, by containing all other classes and implementing the game loop	60
HexMap	A class which defines a hex map of hex tiles	79
HexTile	A class which defines a hex tile of the hex map	102
Message	A structure which defines a standard message format	148
MessageHub	A class which acts as a central hub for inter-object message traffic	149
Settlement	A settlement class (child class of TileImprovement)	157
SolarPV	A settlement class (child class of TileImprovement)	167
TidalTurbine	A settlement class (child class of TileImprovement)	184
TileImprovement	A base class for the tile improvement hierarchy	201
WaveEnergyConverter	A settlement class (child class of TileImprovement)	225
WindTurbine	A settlement class (child class of TileImprovement)	243

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	261
header/ DieselGenerator.h	
Header file for the DieselGenerator class	262
header/ EnergyStorageSystem.h	
Header file for the EnergyStorageSystem class	263
header/ Game.h	290
header/ HexMap.h	
Header file for the HexMap class	291
header/ HexTile.h	
Header file for the Game class	292
header/ Settlement.h	
Header file for the Settlement class	294
header/ SolarPV.h	
Header file for the SolarPV class	295
header/ TidalTurbine.h	
Header file for the TidalTurbine class	296
header/ TileImprovement.h	
Header file for the TileImprovement class	297
header/ WaveEnergyConverter.h	
Header file for the WaveEnergyConverter class	299
header/ WindTurbine.h	
Header file for the WindTurbine class	300
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	264
header/ESC_core/ constants.h	
Header file for various constants	265
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	281
header/ESC_core/ includes.h	
Header file for various includes	282
header/ESC_core/ MessageHub.h	
Header file for the MessageHub class	283
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	283

source/ ContextMenu.cpp	Implementation file for the ContextMenu class	301
source/ DieselGenerator.cpp	Implementation file for the DieselGenerator class	301
source/ EnergyStorageSystem.cpp	Implementation file for the EnergyStorageSystem class	302
source/ Game.cpp	Implementation file for the Game class	309
source/ HexMap.cpp	Implementation file for the HexMap class	309
source/ HexTile.cpp	Implementation file for the HexTile class	310
source/ main.cpp	Implementation file for main() for Road To Zero	310
source/ Settlement.cpp	Implementation file for the Settlement class	315
source/ SolarPV.cpp	Implementation file for the SolarPV class	315
source/ TidalTurbine.cpp	Implementation file for the TidalTurbine class	316
source/ TileImprovement.cpp	Implementation file for the TileImprovement class	316
source/ WaveEnergyConverter.cpp	Implementation file for the WaveEnergyConverter class	316
source/ WindTurbine.cpp	Implementation file for the WindTurbine class	317
source/ESC_core/ AssetsManager.cpp	Implementation file for the AssetsManager class	302
source/ESC_core/ MessageHub.cpp	Implementation file for the MessageHub class	302
source/ESC_core/ testing_utils.cpp	Implementation file for various testing utilities	303

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     }
```



```

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, int, 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 [setIsSelected](#) (bool)
Method to set the is selected attribute.
- void [advanceTurn](#) (void)
Method to handle turn advance.
- 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 exhaust animation).
- int [fuel_cost](#)
The fuel costs for this turn.
- int [emissions_tonnes_CO2e](#)
The emissions for this turn.

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__drawProductionMenu](#) (void)
Helper method to draw production menu assets.
- void [__upgrade](#) (void)
Helper method to upgrade the diesel generator.
- void [__computeProductionCosts](#) (void)
Helper method to compute production costs (fuel, O&M, emissions) based on current production level.
- void [__breakdown](#) (void)
Helper method to trigger an equipment breakdown.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__sendImprovementStateMessage](#) (void)
Helper method to format and sent improvement state message.

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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.

```
469 :
470 TileImprovement (
471     position_x,
472     position_y,
473     tile_resource,
474     event_ptr,
475     render_window_ptr,
476     assets_manager_ptr,
477     message_hub_ptr
478 )
479 {
480     // 1. set attributes
481
482     // 1.1. private
483     //...
484
485     // 1.2. public
486     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
487
488     this->is_running = false;
489
490     this->health = 100;
491
492     this->capacity_kW = 100;
493     this->upgrade_level = 1;
494
495     this->production_MWh = 0;
496     this->max_production_MWh = 72;
497
498     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
499     this->smoke_dx = 5 * SECONDS_PER_FRAME;
500     this->smoke_dy = -10 * SECONDS_PER_FRAME;
501     this->smoke_prob = 16 * SECONDS_PER_FRAME;
502
503     this->smoke_sprite_list = {};
504
505     this->fuel_cost = 0;
506     this->emissions_tonnes_CO2e = 0;
507
508     this->tile_improvement_string = "DIESEL GEN";
509
510     this->__setUpTileImprovementSpriteAnimated();
511
512     std::cout << "DieselGenerator constructed at " << this << std::endl;
513
514     return;
```

```
515 }    /* DieselGenerator() */
```

4.3.2.2 ~DieselGenerator()

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

Destructor for the [DieselGenerator](#) class.

```
897 {
898     std::cout << "DieselGenerator at " << this << " destroyed" << std::endl;
899
900     return;
901 }    /* ~DieselGenerator() */
```

4.3.3 Member Function Documentation

4.3.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
264 {
265     TileImprovement :: __breakdown();
266
267     this->production_MWh = 0;
268     this->fuel_cost = 0;
269     this->operation_maintenance_cost = 0;
270     this->emissions_tonnes_CO2e = 0;
271
272     return;
273 }    /* __breakdown() */
```

4.3.3.2 __computeProductionCosts()

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

Helper method to compute production costs (fuel, O&M, emissions) based on current production level.

```
233 {
234     double litres_diesel = this->production_MWh * LITRES_DIESEL_PER_MWH_PRODUCTION;
235
236     double fuel_cost = (litres_diesel * COST_PER_LITRE_DIESEL) / 1000;
237     this->fuel_cost = round(fuel_cost);
238
239     double emissions_tonnes_CO2e = (litres_diesel * KG_CO2E_PER_LITRE_DIESEL) / 1000;
240     this->emissions_tonnes_CO2e = round(emissions_tonnes_CO2e);
241
242     double operation_maintenance_cost =
243         (this->production_MWh * DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
244     this->operation_maintenance_cost = round(operation_maintenance_cost);
245
246     this->__sendTileStateRequest();
247
248     return;
249 }    /* __computeProductionCosts() */
```

4.3.3.3 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
114 {
115     // 1. draw animated sprite (in off state)
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
127
128         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
129         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
130         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
131     }
132
133     // 2. draw production text
134     std::string production_string = "[W]: INCREASE PRODUCTION\n";
135     production_string += "[S]: DECREASE PRODUCTION\n";
136     production_string += "\n";
137
138     production_string += "PRODUCTION: ";
139     production_string += std::to_string(this->production_MWh);
140     production_string += " MWh (MAX ";
141     production_string += std::to_string(this->max_production_MWh);
142     production_string += ")\n";
143
144     production_string += "FUEL COST: ";
145     production_string += std::to_string(this->fuel_cost);
146     production_string += " K\n";
147
148     production_string += "O&M COST: ";
149     production_string += std::to_string(this->operation_maintenance_cost);
150     production_string += " K\n";
151
152     production_string += "EMISSIONS: ";
153     production_string += std::to_string(this->emissions_tonnes_CO2e);
154     production_string += " tonnes (CO2e)\n";
155
156     sf::Text production_text(
157         production_string,
158         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
159         16
160     );
161
162     production_text.setOrigin(production_text.getLocalBounds().width / 2, 0);
163     production_text.setFillColor(MONOCROME_TEXT_GREEN);
164
165     production_text.setPosition(400 + 30, 400 - 55);
166
167     this->render_window_ptr->draw(production_text);
168
169     return;
170 } /* __drawProductionMenu() */
```

4.3.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
288 {
289     if (this->just_built) {
290         return;
291     }
292 }
```



```

293
294     switch (this->event_ptr->key.code) {
295         case (sf::Keyboard::U): {
296             this->__upgrade();
297
298             break;
299         }
300
301
302         case (sf::Keyboard::W): {
303             if (this->production_menu_open) {
304                 this->production_MWh++;
305
306                 if (this->production_MWh > this->max_production_MWh) {
307                     this->production_MWh = 0;
308                 }
309
310                 this->__computeProductionCosts();
311                 this->assets_manager_ptr->getSound("interface click")->play();
312             }
313
314             break;
315         }
316
317
318         case (sf::Keyboard::S): {
319             if (this->production_menu_open) {
320                 this->production_MWh--;
321
322                 if (this->production_MWh < 0) {
323                     this->production_MWh = this->max_production_MWh;
324                 }
325
326                 this->__computeProductionCosts();
327                 this->assets_manager_ptr->getSound("interface click")->play();
328             }
329
330             break;
331         }
332
333
334         default: {
335             // do nothing!
336
337             break;
338         }
339     }
340
341
342     return;
343 } /* __handleKeyPressEvents() */

```

4.3.3.5 __handleMouseButtonEvents()

```

void DieselGenerator::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

358 {
359     if (this->just_built) {
360         return;
361     }
362
363     switch (this->event_ptr->mouseButton.button) {
364         case (sf::Mouse::Left): {
365             //...
366
367             break;
368         }
369
370
371         case (sf::Mouse::Right): {
372             //...
373
374             break;
375         }
376
377

```

```

378         default: {
379             // do nothing!
380
381             break;
382         }
383     }
384
385     return;
386 } /* __handleMouseButtonEvents() */

```

4.3.3.6 __sendImprovementStateMessage()

```

void DieselGenerator::__sendImprovementStateMessage (
    void ) [private]

```

Helper method to format and sent improvement state message.

```

401 {
402     Message improvement_state_message;
403
404     improvement_state_message.channel = GAME_CHANNEL;
405     improvement_state_message.subject = "improvement state";
406
407     improvement_state_message.int_payload["dispatch_MWh"] = this->production_MWh;
408     improvement_state_message.int_payload["fuel_cost"] = this->fuel_cost;
409     improvement_state_message.int_payload["operation_maintenance_cost"] =
410         this->operation_maintenance_cost;
411     improvement_state_message.int_payload["emissions_tonnes_CO2e"] =
412         this->emissions_tonnes_CO2e;
413
414     this->message_hub_ptr->sendMessage(improvement_state_message);
415
416     std::cout << "Improvement state message sent by " << this << std::endl;
417
418     return;
419 } /* __sendImprovementStateMessage() */

```

4.3.3.7 __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() */

```

4.3.3.8 __upgrade()

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

Helper method to upgrade the diesel generator.

```
185 {
186     if (this->credits < DIESEL_GENERATOR_BUILD_COST) {
187         std::cout << "Cannot upgrade diesel generator: insufficient credits (need "
188             << DIESEL_GENERATOR_BUILD_COST << " K)" << std::endl;
189
190         this->__sendInsufficientCreditsMessage();
191         return;
192     }
193
194     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
195         return;
196     }
197
198     this->is_running = false;
199
200     this->__repair();
201
202     this->capacity_kW += 100;
203     this->upgrade_level++;
204
205     this->production_MWh = 0;
206     this->max_production_MWh += 72;
207
208     this->just_upgraded = true;
209
210     this->assets_manager_ptr->getSound("upgrade")->play();
211
212     this->__sendCreditsSpentMessage(DIESEL_GENERATOR_BUILD_COST);
213     this->__sendTileStateRequest();
214     this->__sendGameStateRequest();
215
216     return;
217 } /* __upgrade() */
```

4.3.3.9 advanceTurn()

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

Method to handle turn advance.

Reimplemented from [TileImprovement](#).

```
625 {
626     // 1. send improvement state message
627     this->__sendImprovementStateMessage();
628
629     // 2. handle start/stop
630     if ((not this->is_running) and (this->production_MWh > 0)) {
631         this->is_running = true;
632         this->assets_manager_ptr->getSound("diesel start")->play();
633     }
634
635     else if (this->is_running and (this->production_MWh <= 0)) {
636         this->is_running = false;
637         this->tile_improvement_sprite_animated[1].setScale(sf::Vector2f(1, 1));
638     }
639
640     // 3. handle equipment health
641     if (this->is_running) {
642         this->health--;
643
644         if (this->health <= 0) {
645             this->__breakdown();
646         }
647     }
648
649     // 4. close menus
650     if (this->production_menu_open) {
```

```

651         this->__closeProductionMenu();
652     }
653
654     if (this->upgrade_menu_open) {
655         this->__closeUpgradeMenu();
656     }
657
658     // 5. send tile state request (if selected)
659     if (this->is_selected) {
660         this->__sendTileStateRequest();
661     }
662
663     return;
664 } /* advanceTurn() */

```

4.3.3.10 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](#).

```

728 {
729     // 1. if just built, call base method and return
730     if (this->just_built) {
731         TileImprovement :: draw();
732     }
733     return;
734 }
735
736 // 2. handle upgrade effects
737 if (this->just_upgraded) {
738     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
739         this->tile_improvement_sprite_animated[i].setColor(
740             sf::Color(
741                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
742                 255,
743                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
744                 255
745             )
746         );
747
748         this->tile_improvement_sprite_animated[i].setScale(
749             sf::Vector2f(
750                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
751                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
752             )
753         );
754     }
755
756     this->upgrade_frame++;
757 }
758
759 if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
760     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
761         this->tile_improvement_sprite_animated[i].setColor(
762             sf::Color(255,255,255,255)
763         );
764
765         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
766     }
767
768     this->just_upgraded = false;
769     this->upgrade_frame = 0;
770 }
771
772 // 3. draw first element of animated sprite
773 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
774
775 // 4. draw second element of animated sprite
776 double move_x = 0;
777 double move_y = 0;
778
779 if (this->is_running) {

```

```

782         this->tile_improvement_sprite_animated[1].setScale(
783             sf::Vector2f(
784                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2),
785                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2)
786             )
787         );
788
789         move_x = 1 * ((double)rand() / RAND_MAX) - 0.5;
790         move_y = 1 * ((double)rand() / RAND_MAX) - 0.5;
791
792         this->tile_improvement_sprite_animated[1].move(move_x, move_y);
793     }
794
795     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
796
797     if (this->is_running) {
798         this->tile_improvement_sprite_animated[1].move(-1 * move_x, -1 * move_y);
799     }
800
801
802     // 5. draw smoke effects
803     if (this->is_running) {
804         if ((double)rand() / RAND_MAX < smoke_prob) {
805             this->smoke_sprite_list.push_back(
806                 sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
807             );
808
809             this->smoke_sprite_list.back().setOrigin(
810                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
811                 this->smoke_sprite_list.back().getLocalBounds().height / 2
812             );
813
814             this->smoke_sprite_list.back().setPosition(
815                 this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
816                 this->position_y - 33
817             );
818         }
819     }
820
821     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
822
823     double alpha = 255;
824
825     while (iter != this->smoke_sprite_list.end()) {
826         this->render_window_ptr->draw(*iter);
827
828         alpha = (*iter).getColor().a;
829
830         alpha -= this->smoke_da;
831
832         if (alpha <= 0) {
833             iter = this->smoke_sprite_list.erase(iter);
834             continue;
835         }
836
837         (*iter).setColor(sf::Color(255, 255, 255, alpha));
838
839         (*iter).move(
840             this->smoke_dx + 2 * ((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
841             this->smoke_dy
842         );
843
844         (*iter).rotate(((double)rand() / RAND_MAX));
845
846         iter++;
847     }
848
849
850     // 6. handle dispatch illustration
851     if (this->production_MWh > 0) {
852         this->dispatch_text.setString(std::to_string(this->production_MWh));
853         this->__drawDispatch();
854     }
855
856
857     // 7. draw production menu
858     if (this->production_menu_open) {
859         this->render_window_ptr->draw(this->production_menu_backing);
860         this->render_window_ptr->draw(this->production_menu_backing_text);
861
862         this->__drawProductionMenu();
863     }
864
865
866     // 8. handle broken effects
867     if (this->is_broken) {
868         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {

```

```

869         this->tile_improvement_sprite_animated[i].setColor(
870             sf::Color(
871                 255,
872                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
873                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
874                 255
875             )
876         );
877     }
878 }
879
880 this->frame++;
881 return;
882 } /* draw() */

```

4.3.3.11 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

532 {
533     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
534
535     // 32 char x 17 line console "-----\n";
536     std::string options_substring = "CAPACITY: ";
537     options_substring += std::to_string(this->capacity_kW);
538     options_substring += " kW (level ";
539     options_substring += std::to_string(this->upgrade_level);
540     options_substring += ")\n";
541
542     options_substring += "PRODUCTION: ";
543     options_substring += std::to_string(this->production_MWh);
544     options_substring += " MWh (MAX ";
545     options_substring += std::to_string(this->max_production_MWh);
546     options_substring += ")\n";
547
548     options_substring += "HEALTH: ";
549     options_substring += std::to_string(this->health);
550     options_substring += "/100";
551
552     if (this->health <= 0) {
553         options_substring += " ** BROKEN! **\n";
554     }
555
556     else {
557         options_substring += "\n";
558     }
559
560     options_substring += "
561     options_substring += " **** DIESEL GEN OPTIONS ****
562     options_substring += "
563
564     options_substring += " [E]: ";
565
566     if (this->is_broken) {
567         options_substring += "*** BROKEN! ***\n";
568     }
569
570     else {
571         options_substring += "OPEN PRODUCTION MENU\n";
572     }
573
574     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
575         options_substring += " [U]: + 100 kW ("
576         options_substring += std::to_string(upgrade_cost);
577         options_substring += " K)\n";
578     }

```

```

579
580     options_substring           += "HOLD [P]: SCRAP (";
581     options_substring           += std::to_string(SCRAP_COST);
582     options_substring           += " K)";
583
584     return options_substring;
585 } /* getTileOptionsSubstring() */

```

4.3.3.12 processEvent()

```

void DieselGenerator::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

679 {
680     TileImprovement :: processEvent ();
681
682     if (this->event_ptr->type == sf::Event::KeyPressed) {
683         this->__handleKeyPressEvents ();
684     }
685
686     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
687         this->__handleMouseButtonEvents ();
688     }
689
690     return;
691 } /* processEvent() */

```

4.3.3.13 processMessage()

```

void DieselGenerator::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

706 {
707     TileImprovement :: processMessage ();
708
709     //...
710
711     return;
712 } /* processMessage() */

```

4.3.3.14 setIsSelected()

```

void DieselGenerator::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](#).

```

602 {
603     TileImprovement :: setIsSelected(is_selected);
604
605     if (this->is_running and this->is_selected) {
606         this->assets_manager_ptr->getSound("diesel running")->play();
607     }
608
609     return;
610 } /* setIsSelected() */

```

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 emissions_tonnes_CO2e

```
int DieselGenerator::emissions_tonnes_CO2e
```

The emissions for this turn.

4.3.4.3 fuel_cost

```
int DieselGenerator::fuel_cost
```

The fuel costs for this turn.

4.3.4.4 max_production_MWh

```
int DieselGenerator::max_production_MWh
```

The maximum production [MWh] for this turn.

4.3.4.5 production_MWh

```
int DieselGenerator::production_MWh
```

The current production [MWh] of the diesel generator.

4.3.4.6 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.7 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.8 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.9 smoke_prob

```
double DieselGenerator::smoke_prob
```

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

4.3.4.10 smoke_sprite_list

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

A list of smoke sprite (for exhaust animation).

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

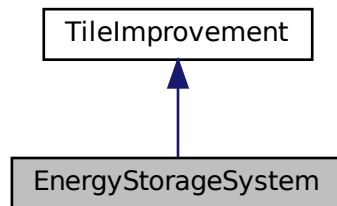
- header/[DieselGenerator.h](#)
- source/[DieselGenerator.cpp](#)

4.4 EnergyStorageSystem Class Reference

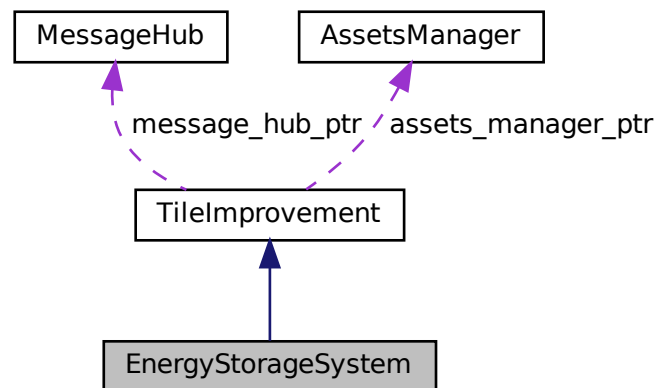
A settlement class (child class of [TileImprovement](#)).

```
#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         default: {
236             // do nothing!
237
238             break;
239         }
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(MONOCROME_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:

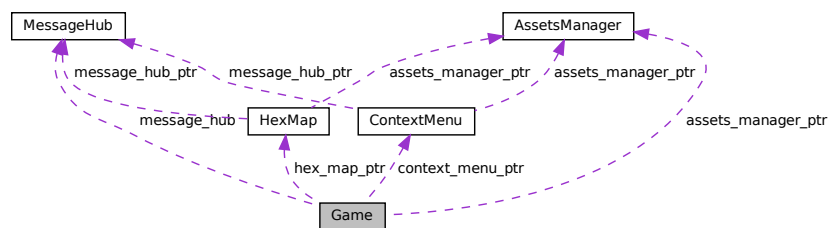
- header/[EnergyStorageSystem.h](#)
- source/[EnergyStorageSystem.cpp](#)

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.
- [bool check_terminating_conditions](#)
Boolean indicating whether or not to check terminating conditions.
- [bool message_deadlock](#)
A boolean indicating whether a message deadlock has been detected.
- [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 demand_remaining_MWh](#)
The current remaining energy demand [MWh].
- [int cumulative_emissions_tonnes](#)
Cumulative emissions [tonnes] (1 tonne = 1000 kg).
- [int message_deadlock_frame](#)
A frame counter for detecting message deadlock.
- [int turn = 0](#)
The current game turn.
- [std::vector< double > demand_vec_MWh](#)
A vector of daily demands [MWh] for the current month.
- [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 [__checkTerminatingConditions](#) (void)
Helper method to check terminating conditions (i.e., loss or victory conditions).
- void [__advanceTurn](#) (void)
Helper method to advance turn.
- void [__computeCurrentDemand](#) (void)
Helper method to compute current energy demand.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__handleImprovementStateMessage](#) (Message)
Helper method to handle improvement state messages.
- 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 [__sendTurnAdvanceMessage](#) (void)
Helper method to format and send a turn advance message.
- void [__sendCreditsEarnedMessage](#) (void)
Helper method to format and send a credits earned 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.

```
924 {
925     // 1. set attributes
926
927     // 1.1. private
928     this->render_window_ptr = render_window_ptr;
929
930     this->assets_manager_ptr = assets_manager_ptr;
931
932     // 1.2. public
933     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
934
935     this->quit_game = false;
936     this->game_loop_broken = false;
937     this->show_frame_clock_overlay = false;
938     this->check_terminating_conditions = false;
939
940     this->frame = 0;
941     this->time_since_start_s = 0;
942
943     this->message_deadlock = false;
944     this->message_deadlock_frame = 0;
945
946     double seconds_since_epoch = time(NULL);
947     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
948
949     this->year = 1970 + (int)years_since_epoch;
950     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
951     while (this->month > 12) {
952         this->month -= 12;
953     }
954
955     this->population = 0;
956     this->credits = STARTING_CREDITS;
957     this->demand_MWh = 0;
958     this->demand_remaining_MWh = 0;
959     this->cumulative_emissions_tonnes = 0;
960
961     this->demand_vec_MWh.resize(30, 0);
962
963     this->hex_map_ptr = new HexMap(
964         6,
965         &(this->event),
966         this->render_window_ptr,
967         this->assets_manager_ptr,
968         &(this->message_hub)
969     );
970
971     this->context_menu_ptr = new ContextMenu(
972         &(this->event),
973         this->render_window_ptr,
974         this->assets_manager_ptr,
975         &(this->message_hub)
976     );
977
978     // 2. add message channel(s)
979     this->message_hub.addChannel(GAME_CHANNEL);
980     this->message_hub.addChannel(GAME_STATE_CHANNEL);
981
982     std::cout << "Game constructed at " << this << std::endl;
983
984     return;
985 } /* Game() */
```

4.5.2.2 ~Game()

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

Destructor for the [Game](#) class.

```
1089 {
1090     // 1. clean up attributes
1091     delete this->hex_map_ptr;
1092     delete this->context_menu_ptr;
1093
1094     std::cout << "Game at " << this << " destroyed" << std::endl;
1095
1096     return;
1097 } /* ~Game() */
```

4.5.3 Member Function Documentation

4.5.3.1 __advanceTurn()

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

Helper method to advance turn.

```
115 {
116     // 1. advance turn
117     this->turn++;
118
119     // 2. advance month/year
120     this->month++;
121     if (this->month > 12) {
122         this->year++;
123         this->month = 1;
124     }
125
126     // 3. update population
127     if (this->turn == 1) {
128         this->population = STARTING_POPULATION;
129     }
130
131     else {
132         this->population = ceil(this->population * POPULATION_MONTHLY_GROWTH_RATE);
133     }
134
135     // 4. update demand
136     this->__computeCurrentDemand();
137
138     // 5. send turn advance message
139     this->__sendTurnAdvanceMessage();
140
141 } /* __advanceTurn() */
```

4.5.3.2 __checkTerminatingConditions()

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

Helper method to check terminating conditions (i.e., loss or victory conditions).

```
94 {
95     std::cout << "Game :: __checkTerminatingConditions()" << std::endl;
96
97     //...
98
99     return;
100 } /* __checkTerminatingConditions() */
```

4.5.3.3 __computeCurrentDemand()

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

Helper method to compute current energy demand.

```
156 {
157     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
158     std::default_random_engine generator(seed);
159
160     std::normal_distribution<double> normal_dist(
161         MEAN_DAILY_DEMAND_RATIOS[this->month - 1],
162         STDEV_DAILY_DEMAND_RATIOS[this->month - 1]
163     );
164
165     double demand_MWh = 0;
166
167     for (int i = 0; i < 30; i++) {
168         this->demand_vec_MWh[i] =
169             normal_dist(generator) * MAXIMUM_DAILY_DEMAND_PER_CAPITA * this->population;
170
171         demand_MWh += this->demand_vec_MWh[i];
172     }
173
174     this->demand_MWh = round(demand_MWh);
175     this->demand_remaining_MWh = this->demand_MWh;
176
177     return;
178 } /* __computeCurrentDemand() */
```

4.5.3.4 __draw()

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

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

```
891 {
892     this->__drawHUD();
893
894     if (this->show_frame_clock_overlay) {
895         this->__drawFrameClockOverlay();
896     }
897
898     return;
899 } /* draw() */
```

4.5.3.5 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
717 {
718     std::string frame_clock_string = "FRAME: ";
719     frame_clock_string += std::to_string(this->frame);
720     frame_clock_string += "\nTIME SINCE START [s]: ";
721     frame_clock_string += std::to_string(this->time_since_start_s);
722
723     sf::Text frame_clock_text(
724         frame_clock_string,
725         *(this->assets_manager_ptr->getFont("DroidSansMono")),
726         16
727     );
728
729     sf::RectangleShape frame_clock_backing(
730         sf::Vector2f(
```

```

731         1.02 * frame_clock_text.getLocalBounds().width,
732         1.20 * frame_clock_text.getLocalBounds().height
733     );
734 };
735 frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
736
737 this->render_window_ptr->draw(frame_clock_backing);
738 this->render_window_ptr->draw(frame_clock_text);
739
740 return;
741 } /* __drawFrameClockOverlay() */

```

4.5.3.6 __drawHUD()

```

void Game::__drawHUD (
    void ) [private]

```

Helper method to heads-up display (HUD).

```

756 {
757     // 1. first line (top)
758     std::string HUD_string = "YEAR: ";
759     HUD_string += std::to_string(this->year);
760
761     HUD_string += "    MONTH: ";
762     HUD_string += std::to_string(this->month);
763
764     HUD_string += "    POPULATION: ";
765     HUD_string += std::to_string(this->population);
766
767     HUD_string += "    CREDITS: ";
768     HUD_string += std::to_string(this->credits);
769     HUD_string += " K";
770
771     HUD_string += "    CURRENT DEMAND: ";
772     HUD_string += std::to_string(this->demand_MWh);
773     HUD_string += " MWh";
774
775     sf::Text HUD_text(
776         HUD_string,
777         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
778         16
779     );
780
781     HUD_text.setPosition(
782         (800 - HUD_text.getLocalBounds().width) / 2,
783         8
784     );
785
786     HUD_text.setFillColor(MONOCROME_TEXT_GREEN);
787
788     this->render_window_ptr->draw(HUD_text);
789
790
791     // 2. second line (top)
792     HUD_string = "CUMULATIVE EMISSIONS: ";
793     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
794     HUD_string += " tonnes (CO2e)";
795
796     HUD_string += "    LIFETIME LIMIT: ";
797     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
798     HUD_string += " tonnes (CO2e)";
799
800     HUD_text.setString(HUD_string);
801
802     HUD_text.setPosition(
803         (800 - HUD_text.getLocalBounds().width) / 2,
804         35
805     );
806
807     this->render_window_ptr->draw(HUD_text);
808
809
810     // 3. third line (bottom)
811     HUD_string = "GAME PHASE: ";
812
813     switch (this->game_phase) {
814         case (GamePhase :: BUILD_SETTLEMENT): {
815             HUD_string += "BUILD SETTLEMENT";

```



```

816
817         break;
818     }
819
820
821     case (GamePhase :: SYSTEM_MANAGEMENT): {
822         HUD_string += "SYSTEM MANAGEMENT";
823
824         break;
825     }
826
827
828     case (GamePhase :: LOSS_EMISSIONS): {
829         HUD_string += "LOSS (EMISSIONS)";
830
831         break;
832     }
833
834
835     case (GamePhase :: LOSS_DEMAND): {
836         HUD_string += "LOSS (DEMAND)";
837
838         break;
839     }
840
841
842     case (GamePhase :: LOSS_CREDITS): {
843         HUD_string += "LOSS (CREDITS)";
844
845         break;
846     }
847
848
849     case (GamePhase :: VICTORY): {
850         HUD_string += "VICTORY";
851
852         break;
853     }
854
855
856     default: {
857         HUD_string += "???";
858
859         break;
860     }
861 }
862
863 HUD_string += "    TURN: ";
864 HUD_string += std::to_string(this->turn);
865
866 HUD_text.setString(HUD_string);
867
868 HUD_text.setPosition(
869     (800 - HUD_text.getLocalBounds().width) / 2,
870     GAME_HEIGHT - 35
871 );
872
873 this->render_window_ptr->draw(HUD_text);
874
875 return;
876 } /* __drawHUD() */

```

4.5.3.7 __handleImprovementStateMessage()

```

void Game::__handleImprovementStateMessage (
    Message improvement_state_message ) [private]

```

Helper method to handle improvement state messages.

```

280 {
281     // 1. unpack message and update game attributes
282     if (improvement_state_message.int_payload.count("dispatch_MWh") > 0) {
283         this->demand_remaining_MWh -= improvement_state_message.int_payload["dispatch_MWh"];
284
285         this->credits +=
286             round(CREDITS_PER_MWH_SERVED * improvement_state_message.int_payload["dispatch_MWh"]);
287
288         this->__sendCreditsEarnedMessage();
289     }

```

```

290
291     if (improvement_state_message.int_payload.count("fuel_cost") > 0) {
292         this->credits -= improvement_state_message.int_payload["fuel_cost"];
293     }
294
295     if (improvement_state_message.int_payload.count("operation_maintenance_cost") > 0) {
296         this->credits -=
297             improvement_state_message.int_payload["operation_maintenance_cost"];
298     }
299
300     if (improvement_state_message.int_payload.count("emissions_tonnes_CO2e") > 0) {
301         this->cumulative_emissions_tonnes +=
302             improvement_state_message.int_payload["emissions_tonnes_CO2e"];
303     }
304
305     return;
306 } /* __handleImprovementStateMessage() */

```

4.5.3.8 __handleKeyPressEvents()

```

void Game::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

193 {
194     switch (this->event.key.code) {
195         case (sf::Keyboard::Enter): {
196             if (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT) {
197                 this->__advanceTurn();
198             }
199
200             break;
201         }
202
203
204         case (sf::Keyboard::Tilde): {
205             this->__toggleFrameClockOverlay();
206
207             break;
208         }
209
210
211         case (sf::Keyboard::Tab): {
212             this->hex_map_ptr->toggleResourceOverlay();
213
214             break;
215         }
216
217
218         default: {
219             // do nothing!
220
221             break;
222         }
223     }
224
225     return;
226 } /* __handleKeyPressEvents() */

```

4.5.3.9 __handleMouseButtonEvents()

```

void Game::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

241 {
242     switch (this->event.mouseButton.button) {
243         case (sf::Mouse::Left): {
244             //...

```

```

245
246         break;
247     }
248
249
250     case (sf::Mouse::Right): {
251         //...
252
253         break;
254     }
255
256
257     default: {
258         // do nothing!
259
260         break;
261     }
262 }
263
264 return;
265 } /* __handleMouseButtonEvents() */

```

4.5.3.10 __insufficientCreditsAlarm()

```

void Game::__insufficientCreditsAlarm (
    void ) [private]

```

Helper method to sound and display and insufficient credits alarm.

```

610 {
611     // 1. sound buzzer
612     this->assets_manager_ptr->getSound("insufficient credits")->play();
613
614     // 2. construct alarm text and backing rectangle
615     sf::Text insufficient_credits_text(
616         "INSUFFICIENT CREDITS",
617         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
618         32
619     );
620
621     insufficient_credits_text.setOrigin(
622         insufficient_credits_text.getLocalBounds().width / 2,
623         insufficient_credits_text.getLocalBounds().height / 2
624     );
625
626     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
627
628     sf::RectangleShape backing_rectangle(
629         sf::Vector2f(
630             1.1 * insufficient_credits_text.getLocalBounds().width,
631             1.5 * insufficient_credits_text.getLocalBounds().height
632         )
633     );
634
635     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
636
637     backing_rectangle.setOrigin(
638         backing_rectangle.getLocalBounds().width / 2,
639         backing_rectangle.getLocalBounds().height / 2
640     );
641
642     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
643
644     // 3. display loop (blocking ~3 seconds)
645     bool red_flag = true;
646     int alarm_frame = 0;
647     double time_since_alarm_s = 0;
648
649     sf::Clock alarm_clock;
650
651     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
652
653         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
654
655         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
656             while (this->render_window_ptr->pollEvent(this->event)) {
657                 // do nothing!
658             }
659

```

```

660
661     this->render_window_ptr->clear();
662
663     this->hex_map_ptr->draw();
664     this->context_menu_ptr->draw();
665     this->__draw();
666
667     if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
668         if (red_flag) {
669             red_flag = false;
670         }
671
672         else {
673             red_flag = true;
674         }
675     }
676
677     if (red_flag) {
678         insufficient_credits_text.setFillColor(MONOCROME_TEXT_RED);
679     }
680
681     else {
682         insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
683     }
684
685     this->render_window_ptr->draw(backing_rectangle);
686     this->render_window_ptr->draw(insufficient_credits_text);
687
688     this->render_window_ptr->display();
689
690     alarm_frame++;
691     this->frame++;
692 }
693
694 // check track status, move to next if stopped
695 if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
696     this->assets_manager_ptr->nextTrack();
697     this->assets_manager_ptr->playTrack();
698 }
699 }
700
701 return;
702 } /* __insufficientCreditsAlarm( */

```

4.5.3.11 __processEvent()

```

void Game::__processEvent (
    void ) [private]

```

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

```

321 {
322     if (this->event.type == sf::Event::Closed) {
323         this->quit_game = true;
324         this->game_loop_broken = true;
325     }
326
327     if (this->event.type == sf::Event::KeyPressed) {
328         this->__handleKeyPressEvents();
329     }
330
331     if (this->event.type == sf::Event::MouseButtonPressed) {
332         this->__handleMouseButtonEvents();
333     }
334
335     return;
336 } /* __processEvent() */

```

4.5.3.12 __processMessage()

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

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

```
490 {
491     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
492         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
493
494         if (game_channel_message.subject == "quit game") {
495             this->quit_game = true;
496             this->game_loop_broken = true;
497
498             std::cout << "Quit game message received by " << this << std::endl;
499             this->message_hub.popMessage(GAME_CHANNEL);
500         }
501
502         if (game_channel_message.subject == "restart game") {
503             this->game_loop_broken = true;
504
505             std::cout << "Restart game message received by " << this << std::endl;
506             this->message_hub.popMessage(GAME_CHANNEL);
507         }
508
509         if (game_channel_message.subject == "state request") {
510             std::cout << "Game state request message received by " << this << std::endl;
511
512             this->__sendGameStateMessage();
513             this->message_hub.popMessage(GAME_CHANNEL);
514         }
515
516         if (game_channel_message.subject == "credits spent") {
517             this->credits -= game_channel_message.int_payload["credits spent"];
518
519             std::cout << "Credits spent message ( " <<
520                 game_channel_message.int_payload["credits spent"] << " ) received by "
521                 << this << std::endl;
522
523             std::cout << "Current credits (Game): " << this->credits << " K" <<
524                 std::endl;
525
526             this->message_hub.popMessage(GAME_CHANNEL);
527         }
528
529         if (game_channel_message.subject == "insufficient credits") {
530             std::cout << "Insufficient credits message received by " << this <<
531                 std::endl;
532
533             this->__insufficientCreditsAlarm();
534
535             this->message_hub.popMessage(GAME_CHANNEL);
536         }
537
538         if (game_channel_message.subject == "update game phase") {
539             std::cout << "Update game phase message received by " << this << std::endl;
540
541             if (
542                 game_channel_message.string_payload["game phase"] == "system management"
543             ) {
544                 this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
545                 this->__advanceTurn();
546             }
547
548             else if (
549                 game_channel_message.string_payload["game phase"] == "loss emissions"
550             ) {
551                 this->game_phase = GamePhase :: LOSS_EMISSIONS;
552             }
553
554             else if (
555                 game_channel_message.string_payload["game phase"] == "loss demand"
556             ) {
557                 this->game_phase = GamePhase :: LOSS_DEMAND;
558             }
559
560             else if (
561                 game_channel_message.string_payload["game phase"] == "loss credits"
562             ) {
563                 this->game_phase = GamePhase :: LOSS_CREDITS;
564             }
565
566             else if (
567                 game_channel_message.string_payload["game phase"] == "victory"
```

```

568         ) {
569             this->game_phase = GamePhase :: VICTORY;
570         }
571
572         this->message_hub.popMessage(GAME_CHANNEL);
573     }
574
575     if (game_channel_message.subject == "improvement state") {
576         std::cout << "Improvement state message received by " << this << std::endl;
577
578         this->__handleImprovementStateMessage(game_channel_message);
579
580         this->message_hub.popMessage(GAME_CHANNEL);
581     }
582 }
583
584 if (not this->message_hub.isEmpty(GAME_STATE_CHANNEL)) {
585     Message game_state_message =
586         this->message_hub.receiveMessage(GAME_STATE_CHANNEL);
587
588     if (game_state_message.subject == "turn advance") {
589         std::cout << "Turn advance message received by " << this << std::endl;
590         this->message_hub.popMessage(GAME_STATE_CHANNEL);
591     }
592 }
593
594 return;
595 } /* __processMessage() */

```

4.5.3.13 __sendCreditsEarnedMessage()

```

void Game::__sendCreditsEarnedMessage (
    void ) [private]

```

Helper method to format and send a credits earned message.

```

465 {
466     Message credits_earned_message;
467
468     credits_earned_message.channel = SETTLEMENT_CHANNEL;
469     credits_earned_message.subject = "credits earned";
470
471     this->message_hub.sendMessage(credits_earned_message);
472
473     std::cout << "Credits earned message sent by " << this << std::endl;
474     return;
475 } /* __sendCreditsEarnedMessage() */

```

4.5.3.14 __sendGameStateMessage()

```

void Game::__sendGameStateMessage (
    void ) [private]

```

Helper method to format and send a game state message.

```

351 {
352     Message game_state_message;
353
354     game_state_message.channel = GAME_STATE_CHANNEL;
355     game_state_message.subject = "game state";
356
357     game_state_message.int_payload["year"] = this->year;
358     game_state_message.int_payload["month"] = this->month;
359     game_state_message.int_payload["population"] = this->population;
360     game_state_message.int_payload["credits"] = this->credits;
361     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
362     game_state_message.int_payload["cumulative_emissions_tonnes"] =
363         this->cumulative_emissions_tonnes;
364
365     switch (this->game_phase) {
366         case (GamePhase :: BUILD_SETTLEMENT): {

```

```

367         game_state_message.string_payload["game phase"] = "build settlement";
368     }
369     break;
370 }
371
372
373     case (GamePhase :: SYSTEM_MANAGEMENT): {
374         game_state_message.string_payload["game phase"] = "system management";
375     }
376     break;
377 }
378
379
380     case (GamePhase :: LOSS_EMISSIONS): {
381         game_state_message.string_payload["game phase"] = "loss emissions";
382     }
383     break;
384 }
385
386
387     case (GamePhase :: LOSS_DEMAND): {
388         game_state_message.string_payload["game phase"] = "loss demand";
389     }
390     break;
391 }
392
393
394     case (GamePhase :: LOSS_CREDITS): {
395         game_state_message.string_payload["game phase"] = "loss credits";
396     }
397     break;
398 }
399
400
401     case (GamePhase :: VICTORY): {
402         game_state_message.string_payload["game phase"] = "victory";
403     }
404     break;
405 }
406
407
408     default: {
409         // do nothing!
410     }
411     break;
412 }
413 }
414
415 game_state_message.vector_payload["demand_vec_MWh"] = this->demand_vec_MWh;
416
417 this->message_hub.sendMessage(game_state_message);
418
419 std::cout << "Game state message sent by " << this << std::endl;
420 return;
421 } /* __sendGameStateMessage() */

```

4.5.3.15 __sendTurnAdvanceMessage()

```

void Game::__sendTurnAdvanceMessage (
    void ) [private]

```

Helper method to format and send a turn advance message.

```

436 {
437     Message turn_advance_message;
438
439     turn_advance_message.channel = GAME_STATE_CHANNEL;
440     turn_advance_message.subject = "turn advance";
441
442     turn_advance_message.int_payload["credits"] = this->credits;
443     turn_advance_message.int_payload["month"] = this->month;
444     turn_advance_message.int_payload["demand_MWh"] = this->demand_MWh;
445
446     this->message_hub.sendMessage(turn_advance_message);
447
448     std::cout << "Turn advance message sent by " << this << std::endl;
449     return;
450 } /* __sendTurnAdvanceMessage() */

```

4.5.3.16 __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() */
```

4.5.3.17 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).

```
1003 {
1004     // 1. play brand animation
1005     //...
1006
1007     // 2. show splash screen
1008     //...
1009
1010     // 3. start game loop
1011     while (not this->game_loop_broken) {
1012         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
1013
1014         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
1015             // 6.1. process events
1016             while (this->render_window_ptr->pollEvent(this->event)) {
1017                 this->hex_map_ptr->processEvent();
1018                 this->context_menu_ptr->processEvent();
1019                 this->__processEvent();
1020             }
1021
1022             // 6.2. process messages
1023             while (this->message_hub.hasTraffic()) {
1024                 this->hex_map_ptr->processMessage();
1025                 this->context_menu_ptr->processMessage();
1026                 this->__processMessage();
1027
1028                 this->check_terminating_conditions = true;
1029
1030                 if (not this->message_deadlock) {
1031                     this->message_deadlock_frame++;
1032
1033                     if (this->message_deadlock_frame > 5 * FRAMES_PER_SECOND) {
1034                         this->message_hub.printState();
1035                         this->message_deadlock = true;
1036                     }
1037                 }
1038             }
1039             this->message_deadlock = false;
1040             this->message_deadlock_frame = 0;
1041
1042             // 6.3. check terminating conditions
1043             if (this->check_terminating_conditions) {
1044                 this->__checkTerminatingConditions();
1045             }
1046         }
```



```

1047         this->check_terminating_conditions = false;
1048     }
1049
1050
1051     // 6.4. draw frame
1052     this->render_window_ptr->clear();
1053
1054     this->hex_map_ptr->draw();
1055     this->context_menu_ptr->draw();
1056     this->__draw();
1057
1058     this->render_window_ptr->display();
1059
1060
1061     // 6.5. increment frame
1062     this->frame++;
1063 }
1064
1065 // check track status, move to next if stopped
1066 if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
1067     this->assets_manager_ptr->nextTrack();
1068     this->assets_manager_ptr->playTrack();
1069 }
1070
1071 }
1072
1073 return this->quit_game;
1074 } /* 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 check_terminating_conditions

`bool Game::check_terminating_conditions`

Boolean indicating whether or not to check terminating conditions.

4.5.4.3 clock

`sf::Clock Game::clock`

The game clock.

4.5.4.4 context_menu_ptr

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

Pointer to the context menu.

4.5.4.5 credits

```
int Game::credits
```

Current balance of credits.

4.5.4.6 cumulative_emissions_tonnes

```
int Game::cumulative_emissions_tonnes
```

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

4.5.4.7 demand_MWh

```
int Game::demand_MWh
```

Current energy demand [MWh].

4.5.4.8 demand_remaining_MWh

```
int Game::demand_remaining_MWh
```

The current remaining energy demand [MWh].

4.5.4.9 demand_vec_MWh

```
std::vector<double> Game::demand_vec_MWh
```

A vector of daily demands [MWh] for the current month.

4.5.4.10 event

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

The game events class.

4.5.4.11 frame

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

The current frame of the game.

4.5.4.12 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.13 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.14 hex_map_ptr

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

Pointer to the hex map (defines game world).

4.5.4.15 message_deadlock

```
bool Game::message_deadlock
```

A boolean indicating whether a message deadlock has been detected.

4.5.4.16 message_deadlock_frame

```
int Game::message_deadlock_frame
```

A frame counter for detecting message deadlock.

4.5.4.17 message_hub

```
MessageHub Game::message_hub
```

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

4.5.4.18 month

```
int Game::month
```

Current game month.

4.5.4.19 population

```
int Game::population
```

Current population.

4.5.4.20 quit_game

```
bool Game::quit_game
```

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

4.5.4.21 render_window_ptr

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

A pointer to the render window.

4.5.4.22 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

4.5.4.23 time_since_start_s

```
double Game::time_since_start_s
```

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

4.5.4.24 turn

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

The current game turn.

4.5.4.25 year

```
int Game::year
```

Current game year.

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

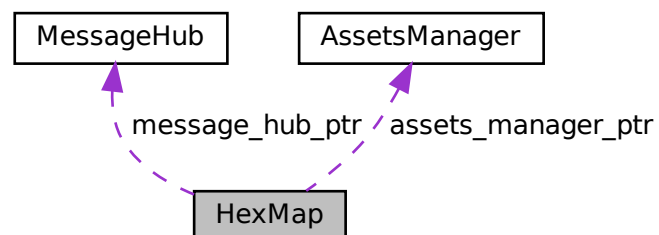
- header/[Game.h](#)
- source/[Game.cpp](#)

4.6 HexMap Class Reference

A class which defines a hex map of hex tiles.

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

Collaboration diagram for HexMap:



Public Member Functions

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

Public Attributes

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

Private Member Functions

- void [__setUpGlassScreen](#) (void)
Helper method to set up glass screen effect (drawable).
- void [__layTiles](#) (void)
Helper method to lay the hex tiles down to generate the game world.
- void [__buildDrawOrderVector](#) (void)
Helper method to build tile drawing order vector.
- std::vector< double > [__getNoise](#) (int, int=128)
Helper method to generate a vector of noise, with values mapped to the closed interval [0, 1]. Applies a random cosine series approach.
- void [__procedurallyGenerateTileTypes](#) (void)
Helper method to procedurally generate tile types and set tiles accordingly.
- std::vector< double > [__getValidMapIndexPositions](#) (double, double)
Helper method to translate given position into valid index position for a.
- std::vector< [HexTile](#) * > [__getNeighboursVector](#) ([HexTile](#) *)
Helper method to assemble a vector pointers to all neighbours of the given tile.
- [TileType](#) [__getMajorityTileType](#) ([HexTile](#) *)
Function to return majority tile type of a tile and its neighbours. If no clear majority, simply returns the type of the given tile.
- void [__smoothTileTypes](#) (void)
Helper method to smooth tile types using a majority rules approach.
- bool [__isLakeTouchingOcean](#) ([HexTile](#) *)
- void [__enforceOceanContinuity](#) (void)
Helper method to scan tiles and enforce ocean continuity. That is to say, if a lake tile is found to be in contact with an ocean tile, then it becomes ocean.
- void [__procedurallyGenerateTileResources](#) (void)
Helper method to procedurally generate tile resources and set tiles accordingly.
- void [__assembleHexMap](#) (void)
Helper method to assemble the hex map.
- [HexTile](#) * [__getSelectedTile](#) (void)
Helper method to get pointer to selected tile.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__sendNoTileSelectedMessage](#) (void)
Helper method to format and send message on no tile selected.
- void [__assessNeighbours](#) ([HexTile](#) *)
Helper method to assess all neighbours of the given tile.

Private Attributes

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

4.6.1 Detailed Description

A class which defines a hex map of hex tiles.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 HexMap()

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

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

Parameters

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

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



```

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

```

4.6.2.2 ~HexMap()

```

HexMap::~~HexMap (
    void )

```

Destructor for the [HexMap](#) class.

```

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

```

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

```

void HexMap::__assembleHexMap (
    void ) [private]

```

Helper method to assemble the hex map.

```

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

```

4.6.3.2 __assessNeighbours()

```

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

```

Helper method to assess all neighbours of the given tile.

Parameters

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

```

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

```

4.6.3.3 __buildDrawOrderVector()

```

void HexMap::__buildDrawOrderVector (
    void ) [private]

```

Helper method to build tile drawing order vector.

```

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

```

4.6.3.4 `__enforceOceanContinuity()`

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

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

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

4.6.3.5 `__getMajorityTileType()`

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

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

Parameters

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

Returns

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

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

```

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

```

4.6.3.6 __getNeighboursVector()

```

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

```

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

Parameters

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

Returns

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

```

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

```

```

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

```

4.6.3.7 __getNoise()

```

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

```

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

Parameters

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

Returns

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

```

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

```

```

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

```

4.6.3.8 __getSelectedTile()

```

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

```

Helper method to get pointer to selected tile.

Returns

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

```

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

```

```

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

```

4.6.3.9 __getValidMapIndexPositions()

```

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

```

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

Parameters

<i>potential_x</i>	The potential x position of the tile.
<i>potential_y</i>	The potential y position of the tile.

Returns

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

```

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

```

```

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

```

4.6.3.10 __handleKeyPressEvents()

```

void HexMap::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

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

```

4.6.3.11 __handleMouseButtonEvents()

```

void HexMap::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

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

```



```

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

```

4.6.3.12 __isLakeTouchingOcean()

```

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

```

4.6.3.13 __layTiles()

```

void HexMap::__layTiles (
    void ) [private]

```

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

```

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

```

```

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

```

```

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

```

4.6.3.14 __procedurallyGenerateTileResources()

```

void HexMap::__procedurallyGenerateTileResources (
    void ) [private]

```

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

```

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

```

```

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

```

4.6.3.15 __procedurallyGenerateTileTypes()

```

void HexMap::__procedurallyGenerateTileTypes (
    void ) [private]

```

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

```

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

```

4.6.3.16 __sendNoTileSelectedMessage()

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

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

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

4.6.3.17 __setUpGlassScreen()

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

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

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

4.6.3.18 __smoothTileTypes()

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

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

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

4.6.3.19 assess()

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

Method to assess the resource of the selected tile.

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

4.6.3.20 clear()

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

Method to clear the hex map.

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

4.6.3.21 draw()

```
void HexMap::draw (
    void )
```

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

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

```

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

```

4.6.3.22 processEvent()

```

void HexMap::processEvent (
    void )

```

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

```

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

```

4.6.3.23 processMessage()

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

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

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

4.6.3.24 reroll()

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

Method to re-roll the hex map.

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

4.6.3.25 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

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



```

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

```

4.6.4 Member Data Documentation

4.6.4.1 assets_manager_ptr

`AssetsManager*` HexMap::assets_manager_ptr [private]

A pointer to the assets manager.

4.6.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

4.6.4.3 event_ptr

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

A pointer to the event class.

4.6.4.4 frame

`unsigned long long int` HexMap::frame

The current frame of this object.

4.6.4.5 glass_screen

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

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

4.6.4.6 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

4.6.4.7 hex_map

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

A position-indexed, nested map of hex tiles.

4.6.4.8 message_hub_ptr

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

A pointer to the message hub.

4.6.4.9 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

4.6.4.10 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

4.6.4.11 position_x

```
double HexMap::position_x
```

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

4.6.4.12 position_y

```
double HexMap::position_y
```

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

4.6.4.13 render_window_ptr

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

A pointer to the render window.

4.6.4.14 show_resource

```
bool HexMap::show_resource
```

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

4.6.4.15 tile_position_x_vec

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

A vector of tile x positions.

4.6.4.16 tile_position_y_vec

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

A vector of tile y position.

4.6.4.17 tile_selected

```
bool HexMap::tile_selected
```

A boolean which indicates if a tile is currently selected.

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

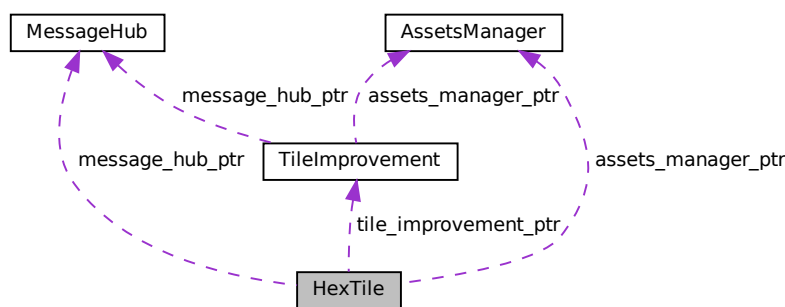
- header/[HexMap.h](#)
- source/[HexMap.cpp](#)

4.7 HexTile Class Reference

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

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

Collaboration diagram for HexTile:



Public Member Functions

- [HexTile](#) (double, double, sf::Event *, sf::RenderWindow *, [AssetsManager](#) *, [MessageHub](#) *)
Constructor for the [HexTile](#) class.
- void [setTileType](#) ([TileType](#))
Method to set the tile type (by enum value).
- void [setTileType](#) (double)
Method to set the tile type (by numeric input).
- void [setTileResource](#) ([TileResource](#))
Method to set the tile resource (by enum value).
- void [setTileResource](#) (double)
Method to set the tile resource (by numeric input).
- void [decorateTile](#) (void)
Method to decorate tile.
- void [toggleResourceOverlay](#) (void)
Method to toggle the tile resource overlay.

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

Public Attributes

- [TileType](#) [tile_type](#)
The terrain type of the tile.
- [TileResource](#) [tile_resource](#)
The renewable resource quality of the tile.
- 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 [__setUpResourceText](#) (void)
- Helper method to set up resource text.*
- void [__setUpMagnifyingGlassSprite](#) (void)
- Helper method to set up and position magnifying glass sprite.*
- void [__setUpTileExplosionReel](#) (void)
- Helper method to set up tile explosion sprite reel.*
- void [__setUpBuildOption](#) (std::string, std::string)
- Helper method to set up and position the sprite and text for a build option.*
- void [__setUpDieselGeneratorBuildOption](#) (void)
- Helper method to set up and position the diesel generator build option.*

- void [__setUpWindTurbineBuildOption](#) (bool=false, bool=false)
Helper method to set up and position the wind turbine build option.
- void [__setUpSolarPVBuildOption](#) (bool=false)
Helper method to set up and position the solar PV array build option.
- void [__setUpTidalTurbineBuildOption](#) (void)
Helper method to set up and position the tidal turbine build option.
- void [__setUpWaveEnergyConverterBuildOption](#) (void)
Helper method to set up and position the wave energy converter build option.
- void [__setUpEnergyStorageSystemBuildOption](#) (void)
Helper method to set up and position the wave energy converter build option.
- void [__setUpBuildMenu](#) (void)
Helper method to set up and place build menu assets (drawable).
- void [__setIsSelected](#) (bool)
Helper method to set the is selected attribute (of tile and improvement).
- void [__clearDecoration](#) (void)
Helper method to clear tile decoration.
- bool [__isClicked](#) (void)
Helper method to determine if tile was clicked on.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__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.

```

2310 {
2311     // 1. set attributes
2312
2313     // 1.1. private
2314     this->event_ptr = event_ptr;
2315     this->render_window_ptr = render_window_ptr;
2316
2317     this->assets_manager_ptr = assets_manager_ptr;
2318     this->message_hub_ptr = message_hub_ptr;
2319
2320     // 1.2. public
2321     this->show_node = false;
2322     this->show_resource = false;
2323     this->resource_assessed = false;
2324     this->resource_assessment = false;
2325     this->is_selected = false;
2326     this->draw_explosion = false;
2327
2328     this->decoration_cleared = false;
2329     this->has_improvement = false;
2330     this->tile_improvement_ptr = NULL;
2331
2332     this->build_menu_open = false;
2333
2334     this->explosion_frame = 0;
2335
2336     this->frame = 0;
2337     this->credits = 0;
2338
2339     this->scrap_improvement_frame = 0;
2340
2341     this->position_x = position_x;
2342     this->position_y = position_y;
2343
2344     this->major_radius = 32;
2345     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2346
2347     this->game_phase = "build settlement";
2348
2349     // 2. set up and position drawable attributes
2350     this->__setUpNodeSprite();
2351     this->__setUpTileSprite();
2352     this->__setUpSelectOutlineSprite();
2353     this->__setUpResourceChipSprite();
2354     this->__setUpResourceText();
2355     this->__setUpMagnifyingGlassSprite();
2356     this->__setUpTileExplosionReel();
2357
2358     // 3. set tile type and resource (default to none type and average)
2359     this->setTileType(TileType :: NONE_TYPE);
2360     this->setTileResource(TileResource :: AVERAGE);
2361
2362     std::cout << "HexTile constructed at " << this << std::endl;
2363
2364     return;
2365 } /* HexTile() */

```

4.7.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

2928 {
2929     if (this->tile_improvement_ptr != NULL) {
2930         delete this->tile_improvement_ptr;
2931     }
2932
2933     std::cout << "HexTile at " << this << " destroyed" << std::endl;
2934
2935     return;
2936 } /* ~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.

```

1411 {
1412     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1413
1414     if (this->credits < build_cost) {
1415         std::cout << "Cannot build diesel generator: insufficient credits (need "
1416             << build_cost << " K)" << std::endl;
1417
1418         this->__sendInsufficientCreditsMessage();
1419         return;
1420     }
1421
1422     this->tile_improvement_ptr = new DieselGenerator(
1423         this->position_x,
1424         this->position_y,
1425         this->tile_resource,
1426         this->event_ptr,
1427         this->render_window_ptr,
1428         this->assets_manager_ptr,
1429         this->message_hub_ptr
1430     );
1431
1432     this->has_improvement = true;
1433     this->__closeBuildMenu();
1434
1435     this->__sendCreditsSpentMessage(build_cost);
1436     this->__sendTileStateMessage();
1437     this->__sendGameStateRequest();
1438
1439     return;
1440 } /* __buildDieselGenerator() */

```

4.7.3.2 __buildEnergyStorage()

```

void HexTile::__buildEnergyStorage (
    void ) [private]

```

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

```

1659 {
1660     /*
1661     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1662
1663     if (this->credits < build_cost) {
1664         std::cout << "Cannot build energy storage system: insufficient credits (need "
1665             << build_cost << " K)" << std::endl;
1666
1667         this->__sendInsufficientCreditsMessage();
1668         return;

```

```

1669     }
1670
1671     this->tile_improvement_ptr = new EnergyStorageSystem(
1672         this->position_x,
1673         this->position_y,
1674         this->event_ptr,
1675         this->render_window_ptr,
1676         this->assets_manager_ptr,
1677         this->message_hub_ptr
1678     );
1679
1680     this->has_improvement = true;
1681     this->__closeBuildMenu();
1682
1683     this->__sendCreditsSpentMessage(build_cost);
1684     this->__sendTileStateMessage();
1685     this->__sendGameStateRequest();
1686     /*
1687     return;
1688     */
1689 } /* __buildEnergyStorage() */

```

4.7.3.3 __buildSettlement()

```

void HexTile::__buildSettlement (
    void ) [private]

```

Helper method to build a settlement on this tile.

```

1364 {
1365     if (this->credits < BUILD_SETTLEMENT_COST) {
1366         std::cout << "Cannot build settlement: insufficient credits (need "
1367             << BUILD_SETTLEMENT_COST << " K)" << std::endl;
1368
1369         this->__sendInsufficientCreditsMessage();
1370         return;
1371     }
1372
1373     this->__clearDecoration();
1374
1375     this->tile_improvement_ptr = new Settlement(
1376         this->position_x,
1377         this->position_y,
1378         this->tile_resource,
1379         this->event_ptr,
1380         this->render_window_ptr,
1381         this->assets_manager_ptr,
1382         this->message_hub_ptr
1383     );
1384
1385     this->has_improvement = true;
1386
1387     this->assess();
1388     this->__sendAssessNeighboursMessage();
1389
1390     this->__sendUpdateGamePhaseMessage("system management");
1391     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1392     this->__sendTileStateMessage();
1393     this->__sendGameStateRequest();
1394
1395     return;
1396 } /* __buildSettlement() */

```

4.7.3.4 __buildSolarPV()

```

void HexTile::__buildSolarPV (
    void ) [private]

```

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

```

1455 {
1456     int build_cost = SOLAR_PV_BUILD_COST;

```

```

1457
1458     if (this->tile_type == TileType :: LAKE) {
1459         build_cost *= SOLAR_PV_WATER_BUILD_MULTIMPLIER;
1460     }
1461
1462     if (this->credits < build_cost) {
1463         std::cout << "Cannot build solar PV array: insufficient credits (need "
1464             << build_cost << " K)" << std::endl;
1465
1466         this->__sendInsufficientCreditsMessage();
1467         return;
1468     }
1469
1470     this->tile_improvement_ptr = new SolarPV(
1471         this->position_x,
1472         this->position_y,
1473         this->tile_resource,
1474         this->event_ptr,
1475         this->render_window_ptr,
1476         this->assets_manager_ptr,
1477         this->message_hub_ptr
1478     );
1479
1480     this->has_improvement = true;
1481     this->__closeBuildMenu();
1482
1483     if (this->tile_type == TileType :: LAKE) {
1484         this->decoration_cleared = true;
1485         this->assets_manager_ptr->getSound("splash")->play();
1486     }
1487
1488     this->__sendCreditsSpentMessage(build_cost);
1489     this->__sendTileStateMessage();
1490     this->__sendGameStateRequest();
1491
1492     return;
1493 } /* __buildSolarPV() */

```

4.7.3.5 __buildTidalTurbine()

```

void HexTile::__buildTidalTurbine (
    void ) [private]

```

Helper method to build a tidal turbine on this tile.

```

1567 {
1568     int build_cost = TIDAL_TURBINE_BUILD_COST;
1569
1570     if (this->credits < build_cost) {
1571         std::cout << "Cannot build tidal turbine: insufficient credits (need "
1572             << build_cost << " K)" << std::endl;
1573
1574         this->__sendInsufficientCreditsMessage();
1575         return;
1576     }
1577
1578     this->tile_improvement_ptr = new TidalTurbine(
1579         this->position_x,
1580         this->position_y,
1581         this->tile_resource,
1582         this->event_ptr,
1583         this->render_window_ptr,
1584         this->assets_manager_ptr,
1585         this->message_hub_ptr
1586     );
1587
1588     this->has_improvement = true;
1589     this->decoration_cleared = true;
1590     this->assets_manager_ptr->getSound("splash")->play();
1591     this->__closeBuildMenu();
1592
1593     this->__sendCreditsSpentMessage(build_cost);
1594     this->__sendTileStateMessage();
1595     this->__sendGameStateRequest();
1596
1597     return;
1598 } /* __buildTidalTurbine() */

```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1613 {
1614     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1615
1616     if (this->credits < build_cost) {
1617         std::cout << "Cannot build wave energy converter: insufficient credits (need "
1618             << build_cost << " K)" << std::endl;
1619
1620         this->__sendInsufficientCreditsMessage();
1621         return;
1622     }
1623
1624     this->tile_improvement_ptr = new WaveEnergyConverter(
1625         this->position_x,
1626         this->position_y,
1627         this->tile_resource,
1628         this->event_ptr,
1629         this->render_window_ptr,
1630         this->assets_manager_ptr,
1631         this->message_hub_ptr
1632     );
1633
1634     this->has_improvement = true;
1635     this->decoration_cleared = true;
1636     this->assets_manager_ptr->getSound("splash")->play();
1637     this->__closeBuildMenu();
1638
1639     this->__sendCreditsSpentMessage(build_cost);
1640     this->__sendTileStateMessage();
1641     this->__sendGameStateRequest();
1642
1643     return;
1644 } /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1508 {
1509     int build_cost = WIND_TURBINE_BUILD_COST;
1510
1511     if (
1512         (this->tile_type == TileType :: LAKE) or
1513         (this->tile_type == TileType :: OCEAN)
1514     ) {
1515         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1516     }
1517
1518     if (this->credits < build_cost) {
1519         std::cout << "Cannot build wind turbine: insufficient credits (need "
1520             << build_cost << " K)" << std::endl;
1521
1522         this->__sendInsufficientCreditsMessage();
1523         return;
1524     }
1525
1526     this->tile_improvement_ptr = new WindTurbine(
1527         this->position_x,
1528         this->position_y,
1529         this->tile_resource,
1530         this->event_ptr,
1531         this->render_window_ptr,
1532         this->assets_manager_ptr,
1533         this->message_hub_ptr
1534     );
1535
1536     this->has_improvement = true;
1537     this->__closeBuildMenu();
```

```

1538
1539     if (
1540         (this->tile_type == TileType :: LAKE) or
1541         (this->tile_type == TileType :: OCEAN)
1542     ) {
1543         this->decoration_cleared = true;
1544         this->assets_manager_ptr->getSound("splash")->play();
1545     }
1546
1547     this->__sendCreditsSpentMessage(build_cost);
1548     this->__sendTileStateMessage();
1549     this->__sendGameStateRequest();
1550
1551     return;
1552 } /* __buildWindTurbine() */

```

4.7.3.8 __clearDecoration()

```

void HexTile::__clearDecoration (
    void ) [private]

```

Helper method to clear tile decoration.

```

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

```

4.7.3.9 __closeBuildMenu()

```

void HexTile::__closeBuildMenu (
    void ) [private]

```

Helper method to close the tile improvement build menu.

```

1339 {
1340     if (not this->build_menu_open) {
1341         return;
1342     }
1343
1344     this->build_menu_open = false;
1345     this->assets_manager_ptr->getSound("build menu close")->play();
1346
1347     return;
1348 } /* __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.

```
1805 {
1806     std::string coords_substring = "TILE COORDS:  ";
1807     coords_substring += std::to_string(int(this->position_x - 400));
1808     coords_substring += ", ";
1809     coords_substring += std::to_string(int(this->position_y - 400));
1810     coords_substring += "\n";
1811
1812     return coords_substring;
1813 } /* __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.

```
1964 {
1965     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1966
1967     if (this->has_improvement) {
1968         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1969         improvement_substring += "\n";
1970     }
1971
1972     else {
1973         improvement_substring += "NONE\n";
1974     }
1975
1976     return improvement_substring;
1977 } /* __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.

```

1994 {
1995     //          32 char x 17 line console "-----\n";
1996     std::string options_substring          = "      **** TILE OPTIONS ****      \n";
1997     options_substring                     += "                                     \n";
1998
1999     if (this->game_phase == "build settlement") {
2000         if (
2001             (this->tile_type != TileType :: OCEAN) and
2002             (this->tile_type != TileType :: LAKE)
2003         ) {
2004             options_substring += "[B]:  BUILD SETTLEMENT (";
2005             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
2006             options_substring += " K)\n";
2007         }
2008     }
2009
2010
2011     else if (this->game_phase == "system management") {
2012         if (this->has_improvement) {
2013             options_substring.clear();
2014             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2015         }
2016
2017
2018         else if (not this->resource_assessed) {
2019             options_substring += "[A]:  ASSESS RESOURCE (";
2020             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2021             options_substring += " K)\n";
2022         }
2023
2024
2025         else if (
2026             (not this->decoration_cleared) and
2027             (this->tile_type != TileType :: OCEAN) and
2028             (this->tile_type != TileType :: LAKE)
2029         ) {
2030             options_substring += "[C]:  CLEAR TILE (";
2031
2032             switch (this->tile_type) {
2033                 case (TileType :: FOREST): {
2034                     options_substring += std::to_string(CLEAR_FOREST_COST);
2035
2036                     break;
2037                 }
2038
2039
2040                 case (TileType :: MOUNTAINS): {
2041                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2042
2043                     break;
2044                 }
2045
2046
2047                 case (TileType :: PLAINS): {
2048                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2049
2050                     break;
2051                 }
2052
2053
2054                 default: {
2055                     //do nothing!
2056
2057                     break;
2058                 }
2059             }
2060
2061             options_substring += " K)\n";
2062         }
2063
2064
2065         else if (
2066             (this->decoration_cleared) or
2067             (this->tile_type == TileType :: OCEAN) or
2068             (this->tile_type == TileType :: LAKE)
2069         ) {
2070             options_substring += "[B]:  OPEN BUILD MENU\n";
2071         }
2072     }
2073
2074
2075     else if (this->game_phase == "victory") {
2076         options_substring          += "      **** VICTORY ****      \n";
2077     }

```



```

2078
2079
2080     else {
2081         options_substring += "        **** LOSS ****        \n";
2082     }
2083
2084     return options_substring;
2085 } /* __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.

```

1894 {
1895     std::string resource_substring = "TILE RESOURCE:        ";
1896
1897     if (this->resource_assessed) {
1898         switch (this->tile_resource) {
1899             case (TileResource :: POOR): {
1900                 resource_substring += "POOR\n";
1901
1902                 break;
1903             }
1904
1905             case (TileResource ::BELOW_AVERAGE): {
1906                 resource_substring += "BELOW AVERAGE\n";
1907
1908                 break;
1909             }
1910
1911             case (TileResource :: AVERAGE): {
1912                 resource_substring += "AVERAGE\n";
1913
1914                 break;
1915             }
1916
1917             case (TileResource :: ABOVE_AVERAGE): {
1918                 resource_substring += "ABOVE AVERAGE\n";
1919
1920                 break;
1921             }
1922
1923             case (TileResource :: GOOD): {
1924                 resource_substring += "GOOD\n";
1925
1926                 break;
1927             }
1928
1929             default: {
1930                 resource_substring += "???\n";
1931
1932                 break;
1933             }
1934         }
1935     }
1936
1937     else {
1938         resource_substring += "???\n";
1939     }
1940
1941     return resource_substring;
1942 } /* __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.

```
1830 {
1831     std::string type_substring = "TILE TYPE:      ";
1832
1833     switch (this->tile_type) {
1834         case (TileType :: FOREST): {
1835             type_substring += "FOREST\n";
1836
1837             break;
1838         }
1839
1840         case (TileType :: LAKE): {
1841             type_substring += "LAKE\n";
1842
1843             break;
1844         }
1845
1846         case (TileType :: MOUNTAINS): {
1847             type_substring += "MOUNTAINS\n";
1848
1849             break;
1850         }
1851
1852         case (TileType :: OCEAN): {
1853             type_substring += "OCEAN\n";
1854
1855             break;
1856         }
1857
1858         case (TileType :: PLAINS): {
1859             type_substring += "PLAINS\n";
1860
1861             break;
1862         }
1863
1864         default: {
1865             type_substring += "???\n";
1866
1867             break;
1868         }
1869     }
1870
1871     return type_substring;
1872 } /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

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

```

969         }
970
971
972         case (sf::Keyboard::W): {
973             this->__buildWindTurbine();
974
975             break;
976         }
977
978
979         case (sf::Keyboard::E): {
980             this->__buildEnergyStorage();
981
982             break;
983         }
984
985
986         default: {
987             // do nothing!
988
989             break;
990         }
991     }
992
993     break;
994 }
995
996
997 case (TileType :: OCEAN): {
998     switch (this->event_ptr->key.code) {
999         case (sf::Keyboard::W): {
1000             this->__buildWindTurbine();
1001
1002             break;
1003         }
1004
1005
1006         case (sf::Keyboard::T): {
1007             this->__buildTidalTurbine();
1008
1009             break;
1010         }
1011
1012
1013         case (sf::Keyboard::A): {
1014             this->__buildWaveEnergyConverter();
1015
1016             break;
1017         }
1018
1019
1020         default: {
1021             // do nothing!
1022
1023             break;
1024         }
1025     }
1026
1027     break;
1028 }
1029
1030
1031 case (TileType :: PLAINS): {
1032     switch (this->event_ptr->key.code) {
1033         case (sf::Keyboard::D): {
1034             this->__buildDieselGenerator();
1035
1036             break;
1037         }
1038
1039
1040         case (sf::Keyboard::S): {
1041             this->__buildSolarPV();
1042
1043             break;
1044         }
1045
1046
1047         case (sf::Keyboard::W): {
1048             this->__buildWindTurbine();
1049
1050             break;
1051         }
1052
1053
1054         case (sf::Keyboard::E): {
1055             this->__buildEnergyStorage();

```

```

1056
1057             break;
1058         }
1059
1060         default: {
1061             // do nothing!
1062
1063             break;
1064         }
1065     }
1066 }
1067
1068 break;
1069 }
1070
1071
1072 default: {
1073     //do nothing!
1074
1075     break;
1076 }
1077 }
1078 }
1079
1080
1081 if (this->game_phase == "build settlement") {
1082     if (
1083         (this->tile_type != TileType :: OCEAN) and
1084         (this->tile_type != TileType :: LAKE)
1085     ) {
1086         if (this->event_ptr->key.code == sf::Keyboard::B) {
1087             this->__buildSettlement();
1088         }
1089     }
1090 }
1091
1092
1093 else if (this->game_phase == "system management") {
1094     if (this->has_improvement) {
1095         if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
1096     {
1097         if (this->event_ptr->key.code == sf::Keyboard::P) {
1098             this->__scrapImprovement();
1099         }
1100     }
1101
1102     /*
1103     * All other inputs will be caught and handled by
1104     * this->tile_improvement_ptr->processEvent()
1105     */
1106 }
1107
1108 else if (not this->resource_assessed) {
1109     if (this->event_ptr->key.code == sf::Keyboard::A) {
1110         if (this->credits < RESOURCE_ASSESSMENT_COST) {
1111             std::cout << "Cannot assess resource: insufficient credits (need "
1112                 << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
1113
1114             this->__sendInsufficientCreditsMessage();
1115         }
1116
1117         else {
1118             this->assess();
1119             this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1120             this->__sendTileStateMessage();
1121             this->__sendGameStateRequest();
1122         }
1123     }
1124 }
1125
1126
1127 else if (
1128     (not this->decoration_cleared) and
1129     (this->tile_type != TileType :: OCEAN) and
1130     (this->tile_type != TileType :: LAKE)
1131 ) {
1132     if (this->event_ptr->key.code == sf::Keyboard::C) {
1133         int clear_cost = 0;
1134
1135         switch (this->tile_type) {
1136             case (TileType :: FOREST): {
1137                 clear_cost = CLEAR_FOREST_COST;
1138
1139                 break;
1140             }
1141

```

```

1142
1143         case (TileType :: MOUNTAINS): {
1144             clear_cost = CLEAR_MOUNTAINS_COST;
1145
1146             break;
1147         }
1148
1149
1150         case (TileType :: PLAINS): {
1151             clear_cost = CLEAR_PLAINS_COST;
1152
1153             break;
1154         }
1155
1156
1157         default: {
1158             // do nothing!
1159
1160             break;
1161         }
1162     }
1163
1164     if (this->credits < clear_cost) {
1165         std::cout << "Cannot clear tile: insufficient credits (need "
1166             << clear_cost << " K)" << std::endl;
1167
1168         this->__sendInsufficientCreditsMessage();
1169     }
1170
1171     else {
1172         this->__clearDecoration();
1173         this->__sendCreditsSpentMessage(clear_cost);
1174         this->__sendTileStateMessage();
1175         this->__sendGameStateRequest();
1176     }
1177 }
1178
1179
1180
1181     else if (
1182         (this->decoration_cleared) or
1183         (this->tile_type == TileType :: OCEAN) or
1184         (this->tile_type == TileType :: LAKE)
1185     ) {
1186         if (this->event_ptr->key.code == sf::Keyboard::B) {
1187             this->__openBuildMenu();
1188         }
1189     }
1190 }
1191
1192 return;
1193 } /* __handleKeyPressEvents() */

```

4.7.3.16 __handleKeyReleaseEvents()

```

void HexTile::__handleKeyReleaseEvents (
    void ) [private]
1199 {
1200     if (not this->is_selected) {
1201         return;
1202     }
1203
1204
1205     switch (this->event_ptr->key.code) {
1206         case (sf::Keyboard::P): {
1207             if (this->has_improvement) {
1208                 this->scrap_improvement_frame = 0;
1209
1210                 if (
1211                     this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1212                 ) {
1213                     this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1214                         sf::Color(255, 255, 255, 255)
1215                     );
1216                 }
1217
1218                 else {
1219                     for (
1220                         size_t i = 0;

```

```

1221             i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1222             i++;
1223         } {
1224             this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1225                 sf::Color(255, 255, 255, 255)
1226             );
1227         }
1228     }
1229 }
1230
1231
1232     break;
1233 }
1234
1235
1236     default: {
1237         // do nothing!
1238
1239         break;
1240     }
1241 }
1242
1243 /*
1244 if (this->event_ptr->key.code == sf::Keyboard::P) {
1245 }
1246 */
1247 */
1248
1249 return;
1250 } /* __handleKeyReleaseEvents() */

```

4.7.3.17 __handleMouseButtonEvents()

```

void HexTile::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

1263 {
1264     switch (this->event_ptr->mouseButton.button) {
1265         case (sf::Mouse::Left): {
1266             if (this->__isClicked()) {
1267                 std::cout << "Tile (" << this->position_x << ", " <<
1268                     this->position_y << ") was selected" << std::endl;
1269
1270                 this->__setIsSelected(true);
1271
1272                 this->__sendTileSelectedMessage();
1273                 this->__sendTileStateMessage();
1274                 this->__sendGameStateRequest();
1275             }
1276
1277             else {
1278                 this->__setIsSelected(false);
1279             }
1280
1281             break;
1282         }
1283
1284         case (sf::Mouse::Right): {
1285             this->__setIsSelected(false);
1286
1287             break;
1288         }
1289
1290         default: {
1291             // do nothing!
1292
1293             break;
1294         }
1295     }
1296
1297     return;
1298 }
1299
1300 } /* __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.

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

4.7.3.19 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

```
1315 {
1316     if (this->build_menu_open) {
1317         return;
1318     }
1319
1320     this->build_menu_open = true;
1321     this->assets_manager_ptr->getSound("build menu open")->play();
1322
1323     return;
1324 } /* __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.

```
1704 {
1705     // 1. implement key hold confirmation
1706     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1707         double colour_scalar =
1708             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1709
1710         if (
1711             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1712         ) {
```



```

1713         this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1714             sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1715         );
1716     }
1717
1718     else {
1719         for (
1720             size_t i = 0;
1721             i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1722             i++
1723         ) {
1724             this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1725                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1726             );
1727         }
1728     }
1729
1730     this->scrap_improvement_frame += 4;
1731 }
1732
1733
1734 // 2. carry out scrapping
1735 else {
1736     this->draw_explosion = true;
1737     this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1738
1739     if (this->tile_improvement_ptr->production_menu_open) {
1740         this->tile_improvement_ptr->production_menu_open = false;
1741         this->assets_manager_ptr->getSound("build menu close")->play();
1742     }
1743
1744     delete this->tile_improvement_ptr;
1745     this->tile_improvement_ptr = NULL;
1746
1747     this->has_improvement = false;
1748
1749     this->scrap_improvement_frame = 0;
1750
1751     if (
1752         (this->tile_type == TileType :: LAKE) or
1753         (this->tile_type == TileType :: OCEAN)
1754     ) {
1755         this->decoration_cleared = false;
1756     }
1757
1758     this->__sendCreditsSpentMessage(SCRAP_COST);
1759     this->__sendTileStateMessage();
1760     this->__sendGameStateRequest();
1761 }
1762
1763 return;
1764 } /* __scrapImprovement() */

```

4.7.3.21 __sendAssessNeighboursMessage()

```

void HexTile::__sendAssessNeighboursMessage (
    void ) [private]

```

Helper method to format and send assess neighbours message.

```

2141 {
2142     Message assess_neighbours_message;
2143
2144     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2145     assess_neighbours_message.subject = "assess neighbours";
2146
2147     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2148
2149     std::cout << "Assess neighbours message sent by " << this << std::endl;
2150
2151     return;
2152 } /* __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.
----------------------	--

```
2224 {
2225     Message credits_spent_message;
2226
2227     credits_spent_message.channel = GAME_CHANNEL;
2228     credits_spent_message.subject = "credits spent";
2229
2230     credits_spent_message.int_payload["credits spent"] = credits_spent;
2231
2232     this->message_hub_ptr->sendMessage(credits_spent_message);
2233
2234     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2235         << std::endl;
2236     return;
2237 } /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2167 {
2168     Message game_state_request;
2169
2170     game_state_request.channel = GAME_CHANNEL;
2171     game_state_request.subject = "state request";
2172
2173     this->message_hub_ptr->sendMessage(game_state_request);
2174
2175     std::cout << "Game state request message sent by " << this << std::endl;
2176     return;
2177 } /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2252 {
2253     Message insufficient_credits_message;
2254
2255     insufficient_credits_message.channel = GAME_CHANNEL;
2256     insufficient_credits_message.subject = "insufficient credits";
2257
2258     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2259
2260     std::cout << "Insufficient credits message sent by " << this << std::endl;
2261
2262     return;
2263 } /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1779 {
1780     Message tile_selected_message;
1781
1782     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1783     tile_selected_message.subject = "tile selected";
1784
1785     this->message_hub_ptr->sendMessage(tile_selected_message);
1786
1787     return;
1788 } /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2100 {
2101     Message tile_state_message;
2102
2103     tile_state_message.channel = TILE_STATE_CHANNEL;
2104     tile_state_message.subject = "tile state";
2105
2106
2107     //          32 char x 17 line console "-----\n";
2108     std::string console_string = "      **** TILE INFO ****      \n";
2109
2110     console_string += this->__getTileCoordsSubstring();
2111     console_string += "      \n";
2112
2113     console_string += this->__getTileTypeSubstring();
2114     console_string += this->__getTileResourceSubstring();
2115     console_string += this->__getTileImprovementSubstring();
2116     console_string += "      \n";
2117
2118     console_string += this->__getTileOptionsSubstring();
2119
2120     tile_state_message.string_payload["console string"] = console_string;
2121
2122     this->message_hub_ptr->sendMessage(tile_state_message);
2123
2124     std::cout << "Tile state message sent by " << this << std::endl;
2125     return;
2126 } /* __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.
-------------------	-------------------------

```

2194 {
2195     Message update_game_phase_message;
2196
2197     update_game_phase_message.channel = GAME_CHANNEL;
2198     update_game_phase_message.subject = "update game phase";
2199
2200     update_game_phase_message.string_payload["game phase"] = game_phase;
2201
2202     this->message_hub_ptr->sendMessage(update_game_phase_message);
2203
2204     std::cout << "Update game phase message sent by " << this << std::endl;
2205
2206     return;
2207 } /* __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         this->tile_improvement_ptr->update();
770     }
771
772     if ((not is_selected) and this->build_menu_open) {
773         this->__closeBuildMenu();
774     }
775
776     return;
777 } /* __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(MONOCROME_TEXT_RED);
203
204                 break;
205             }
206
207             case (TileResource :: BELOW_AVERAGE): {
208                 this->resource_text.setString("-1");
209                 this->resource_text.setFillColor(MONOCROME_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                += " \n";
643     energy_storage_system_string                += "CAPCTY:   1 MWh\n";
644     energy_storage_system_string                += "COST:      ";
645     energy_storage_system_string                += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
646     energy_storage_system_string                += " K\n\n";
647     energy_storage_system_string                += "BUILD:      [E]   \n";
648
649     // 3. call general method
650     this->__setUpBuildOption(texture_key, energy_storage_system_string);
651     */
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             this->explosion_sprite_reel.back().setOrigin(
318                 this->explosion_sprite_reel.back().getLocalBounds().width / 2,
319                 this->explosion_sprite_reel.back().getLocalBounds().height / 2
320             );
321             this->explosion_sprite_reel.back().setPosition(
322                 this->position_x,
323                 this->position_y
324             );
325         }
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     // -----\n"
606     std::string wave_energy_converter_string = "WAVE ENERGY CVTR\n";
607     wave_energy_converter_string += " \n";
608     wave_energy_converter_string += "CAPACITY: 100 kW\n";
609     wave_energy_converter_string += "COST: ";
610     wave_energy_converter_string += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
611     wave_energy_converter_string += " K\n\n";
612     wave_energy_converter_string += "BUILD: [A] \n";
613
614     // 3. call general method
615     this->__setUpBuildOption(texture_key, wave_energy_converter_string);
616
617     return;
618 } /* __setUpWaveEnergyConverterBuildOption() */

```

4.7.3.43 __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     // -----\n"
481     std::string wind_turbine_string = " WIND TURBINE \n";
482     wind_turbine_string += " \n";
483     wind_turbine_string += "CAPACITY: 100 kW\n";
484     wind_turbine_string += "COST: ";
485     wind_turbine_string += std::to_string(build_cost);
486     wind_turbine_string += " K";
487
488     if (is_lake) {
489         wind_turbine_string += "\n** LAKE BUILD **\n\n";
490     }
491     else if (is_ocean) {
492         wind_turbine_string += "\n* OCEAN BUILD * \n\n";
493     }
494     else {
495         wind_turbine_string += "\n\n\n";
496     }
497
498     wind_turbine_string += "BUILD: [W] \n";
499
500     // 3. call general method
501     this->__setUpBuildOption(texture_key, wind_turbine_string);
502
503     return;
504 } /* __setUpWindTurbineBuildOption() */

```

4.7.3.44 assess()

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

Method to assess the tile's resource.

```
2686 {
2687     this->resource_assessed = true;
2688     this->resource_assessment = true;
2689
2690     this->assets_manager_ptr->getSound("resource assessment")->play();
2691
2692     this->__setResourceText();
2693     this->__sendTileStateMessage();
2694
2695     return;
2696 } /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2564 {
2565     switch (this->tile_type) {
2566     case (TileType :: FOREST): {
2567         this->tile_decoration_sprite.setTexture(
2568             *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2569         );
2570
2571         break;
2572     }
2573
2574     case (TileType :: LAKE): {
2575         this->tile_decoration_sprite.setTexture(
2576             *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2577         );
2578
2579         break;
2580     }
2581
2582     case (TileType :: MOUNTAINS): {
2583         this->tile_decoration_sprite.setTexture(
2584             *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2585         );
2586
2587         break;
2588     }
2589
2590     case (TileType :: OCEAN): {
2591         this->tile_decoration_sprite.setTexture(
2592             *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2593         );
2594
2595         break;
2596     }
2597
2598     case (TileType :: PLAINS): {
2599         this->tile_decoration_sprite.setTexture(
2600             *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2601         );
2602
2603         break;
2604     }
2605
2606     default: {
2607         // do nothing!
2608
2609         break;
2610     }
2611 }
2612
2613
2614 if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```

2615         this->tile_decoration_sprite.setOrigin(
2616             this->tile_decoration_sprite.getLocalBounds().width / 2,
2617             this->tile_decoration_sprite.getLocalBounds().height / 2
2618         );
2619
2620         this->tile_decoration_sprite.setPosition(
2621             this->position_x,
2622             this->position_y
2623         );
2624
2625         if ((double)rand() / RAND_MAX > 0.5) {
2626             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2627         }
2628     }
2629
2630     else {
2631         this->tile_decoration_sprite.setOrigin(
2632             this->tile_decoration_sprite.getLocalBounds().width / 2,
2633             this->tile_decoration_sprite.getLocalBounds().height
2634         );
2635
2636         this->tile_decoration_sprite.setPosition(
2637             this->position_x,
2638             this->position_y + 12
2639         );
2640
2641         if ((double)rand() / RAND_MAX > 0.5) {
2642             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2643         }
2644     }
2645
2646     return;
2647 } /* 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.

```

2822 {
2823     // 1. draw hex
2824     this->render_window_ptr->draw(this->tile_sprite);
2825
2826     // 2. draw node
2827     if (this->show_node) {
2828         this->render_window_ptr->draw(this->node_sprite);
2829     }
2830
2831     // 3. draw tile decoration
2832     if (not this->decoration_cleared) {
2833         this->render_window_ptr->draw(this->tile_decoration_sprite);
2834     }
2835
2836     // 4. draw selection outline
2837     if (this->is_selected) {
2838         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2839
2840         outline_colour.a =
2841             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2842
2843         this->select_outline_sprite.setOutlineColor(outline_colour);
2844
2845         this->render_window_ptr->draw(this->select_outline_sprite);
2846     }
2847
2848     // 5. draw tile improvement
2849     if (this->has_improvement) {
2850         if (not this->tile_improvement_ptr->just_built) {
2851             this->tile_improvement_ptr->draw();
2852         }
2853     }
2854
2855     // 6. draw resource
2856     if (this->show_resource) {
2857         this->render_window_ptr->draw(this->resource_chip_sprite);
2858         this->render_window_ptr->draw(this->resource_text);
2859     }

```

```

2860
2861 // 7. draw resource assessment notification
2862 if (this->resource_assessment) {
2863     int alpha = this->magnifying_glass_sprite.getColor().a;
2864
2865     alpha -= 0.05 * FRAMES_PER_SECOND;
2866     if (alpha < 0) {
2867         alpha = 0;
2868         this->resource_assessment = false;
2869     }
2870
2871     this->magnifying_glass_sprite.setColor(
2872         sf::Color(255, 255, 255, alpha)
2873     );
2874
2875     this->render_window_ptr->draw(this->magnifying_glass_sprite);
2876 }
2877
2878 // 8. draw explosion, then settlement placement
2879 if (this->draw_explosion) {
2880     this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2881
2882     if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2883         this->explosion_frame++;
2884     }
2885
2886     if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2887         this->draw_explosion = false;
2888         this->explosion_frame = 0;
2889     }
2890 }
2891
2892 else if (this->has_improvement) {
2893     if (this->tile_improvement_ptr->just_built) {
2894         this->tile_improvement_ptr->draw();
2895     }
2896 }
2897
2898 // 9. build menu
2899 if (this->build_menu_open) {
2900     this->render_window_ptr->draw(this->build_menu_backing);
2901     this->render_window_ptr->draw(this->build_menu_backing_text);
2902
2903     for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2904         for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2905             this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2906         }
2907         this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2908     }
2909 }
2910
2911 this->frame++;
2912 return;
2913 } /* draw() */

```

4.7.3.47 processEvent()

```

void HexTile::processEvent (
    void )

```

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

```

2711 {
2712     // 1. process TileImprovement events
2713     if (
2714         this->is_selected and
2715         this->tile_improvement_ptr != NULL
2716     ) {
2717         this->tile_improvement_ptr->processEvent();
2718     }
2719
2720     // 2. process HexTile events
2721     if (this->event_ptr->type == sf::Event::KeyPressed) {
2722         this->__handleKeyPressEvents();
2723     }
2724
2725     if (this->event_ptr->type == sf::Event::KeyReleased) {
2726         this->__handleKeyReleaseEvents();
2727     }

```

```

2728
2729     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2730         this->__handleMouseButtonEvents();
2731     }
2732
2733     return;
2734 } /* processEvent() */

```

4.7.3.48 processMessage()

```

void HexTile::processMessage (
    void )

```

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

```

2749 {
2750     // 1. process TileImprovement messages
2751     if (this->tile_improvement_ptr != NULL) {
2752         this->tile_improvement_ptr->processMessage();
2753     }
2754
2755     // 2. process HexTile messages
2756     if (this->is_selected) {
2757         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2758             Message game_state_message = this->message_hub_ptr->receiveMessage(
2759                 GAME_STATE_CHANNEL
2760             );
2761
2762             if (game_state_message.subject == "game state") {
2763                 this->credits = game_state_message.int_payload["credits"];
2764                 this->game_phase = game_state_message.string_payload["game phase"];
2765
2766                 if (this->tile_improvement_ptr != NULL) {
2767                     this->tile_improvement_ptr->credits = this->credits;
2768                     this->tile_improvement_ptr->game_phase = this->game_phase;
2769
2770                     this->tile_improvement_ptr->month =
2771                         game_state_message.int_payload["month"];
2772
2773                     this->tile_improvement_ptr->demand_MWh =
2774                         game_state_message.int_payload["demand_MWh"];
2775
2776                     this->tile_improvement_ptr->demand_vec_MWh =
2777                         game_state_message.vector_payload["demand_vec_MWh"];
2778
2779                     this->tile_improvement_ptr->update();
2780                 }
2781
2782                 std::cout << "Game state message received by " << this << std::endl;
2783                 this->__sendTileStateMessage();
2784                 this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
2785             }
2786         }
2787
2788         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2789             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2790                 TILE_STATE_CHANNEL
2791             );
2792
2793             if (tile_state_message.subject == "state request") {
2794                 this->__sendTileStateMessage();
2795
2796                 std::cout << "Tile state request received by " << this << std::endl;
2797                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2798             }
2799         }
2800
2801         std::cout << "Current credits (HexTile): " << this->credits << " K" <<
2802             std::endl;
2803     }
2804
2805     return;
2806 } /* processMessage() */

```


4.7.3.49 setTitleResource() [1/2]

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

```
2513 {
2514     // 1. check input
2515     if (input_value < 0 or input_value > 1) {
2516         std::string error_str = "ERROR HexTile::setTitleResource() given input value is ";
2517         error_str += "not in the closed interval [0, 1]";
2518
2519         #ifdef _WIN32
2520             std::cout << error_str << std::endl;
2521         #endif /* _WIN32 */
2522
2523         throw std::runtime_error(error_str);
2524     }
2525
2526     // 2. convert input value to tile resource
2527     TileResource tile_resource;
2528
2529     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2530         tile_resource = TileResource :: POOR;
2531     }
2532     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2533         tile_resource = TileResource :: BELOW_AVERAGE;
2534     }
2535     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2536         tile_resource = TileResource :: AVERAGE;
2537     }
2538     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2539         tile_resource = TileResource :: ABOVE_AVERAGE;
2540     }
2541     else {
2542         tile_resource = TileResource :: GOOD;
2543     }
2544
2545     // 3. call alternate method
2546     this->setTitleResource(tile_resource);
2547
2548     return;
2549 } /* setTitleResource(double) */
```

4.7.3.50 setTitleResource() [2/2]

```
void HexTile::setTitleResource (
    TileResource tile_resource )
```

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

Parameters

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

```
2491 {
2492     this->tile_resource = tile_resource;
2493     this->__setResourceText();
2494
2495     return;
2496 } /* setTitleResource(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].
--------------------	--

```
2441 {
2442     // 1. check input
2443     if (input_value < 0 or input_value > 1) {
2444         std::string error_str = "ERROR HexTile::setTileType() given input value is ";
2445         error_str += "not in the closed interval [0, 1]";
2446
2447         #ifdef _WIN32
2448             std::cout << error_str << std::endl;
2449         #endif /* _WIN32 */
2450
2451         throw std::runtime_error(error_str);
2452     }
2453
2454     // 2. convert input value to tile type
2455     TileType tile_type;
2456
2457     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2458         tile_type = TileType :: LAKE;
2459     }
2460     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2461         tile_type = TileType :: PLAINS;
2462     }
2463     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2464         tile_type = TileType :: FOREST;
2465     }
2466     else {
2467         tile_type = TileType :: MOUNTAINS;
2468     }
2469
2470     // 3. call alternate method
2471     this->setTileType(tile_type);
2472
2473     return;
2474 } /* 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.
------------------	---

```
2380 {
2381     this->tile_type = tile_type;
2382
2383     switch (this->tile_type) {
2384         case (TileType :: FOREST): {
2385             this->tile_sprite.setFillColor(FOREST_GREEN);
2386
2387             break;
2388         }
2389
2390         case (TileType :: LAKE): {
```

```

2391         this->tile_sprite.setFillColor(LAKE_BLUE);
2392
2393         break;
2394     }
2395
2396     case (TileType :: MOUNTAINS): {
2397         this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2398
2399         break;
2400     }
2401
2402     case (TileType :: OCEAN): {
2403         this->tile_sprite.setFillColor(OCEAN_BLUE);
2404
2405         break;
2406     }
2407
2408     case (TileType :: PLAINS): {
2409         this->tile_sprite.setFillColor(PLAINS_YELLOW);
2410
2411         break;
2412     }
2413
2414     default: {
2415         // do nothing!
2416
2417         break;
2418     }
2419 }
2420
2421 this->__setUpBuildMenu();
2422
2423 return;
2424 } /* setTileType(TileType) */

```

4.7.3.53 toggleResourceOverlay()

```

void HexTile::toggleResourceOverlay (
    void )

```

Method to toggle the tile resource overlay.

```

2662 {
2663     if (this->show_resource) {
2664         this->show_resource = false;
2665     }
2666     else {
2667         this->show_resource = true;
2668     }
2669
2670     return;
2671 } /* 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
```

The renewable resource quality of the tile.

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 terrain type of the tile.

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 = {}`
An int payload.
- `std::map< std::string, double > double_payload = {}`
A double payload.
- `std::map< std::string, std::vector< double > > vector_payload = {}`
A vector (double) 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 double payload.

4.8.2.4 int_payload

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

An int 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.

4.8.2.7 vector_payload

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

A vector (double) payload.

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 [printStats](#) (void)
Method for printing message hub state information (mostly for troubleshooting message deadlocks).
- 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.

```
476 {
477     this->clear();
478
479     std::cout << "MessageHub at " << this << " destroyed" << std::endl;
480
481     return;
482 } /* ~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](#).

```

456 {
457
458     this->clearMessages();
459     this->message_map.clear();
460
461     return;
462 } /* clear() */

```

4.9.3.3 clearMessages()

```

void MessageHub::clearMessages (
    void )

```

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

```

430 {
431     std::map<std::string, std::list<Message>::iterator map_iter;
432     for (
433         map_iter = this->message_map.begin();
434         map_iter != this->message_map.end();
435         map_iter++
436     ) {
437         map_iter->second.clear();
438     }
439
440     return;
441 } /* clearMessages() */

```

4.9.3.4 hasTraffic()

```

bool MessageHub::hasTraffic (
    void )

```

Method to determine if there remains any message traffic.

```

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

```

4.9.3.5 isEmpty()

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

Method to check if channel is empty.

Parameters

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

Returns

A boolean indicating whether the channel is empty or not.

```

295 {
296     // 1. check if channel is in map (if not, throw error)
297     if (this->message_map.count(channel) <= 0) {
298         std::string error_str = "ERROR MessageHub::isEmpty() channel ";
299         error_str += channel;
300         error_str += " is not in message map";
301
302         #ifdef _WIN32
303             std::cout << error_str << std::endl;
304         #endif /* _WIN32 */
305
306         throw std::runtime_error(error_str);
307     }
308
309     if (this->message_map[channel].empty()) {
310         return true;
311     }
312     else {
313         return false;
314     }
315 } /* 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.
----------------	---

```

384 {
385     // 1. check if channel is in map (if not, throw error)
386     if (this->message_map.count(channel) <= 0) {
387         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
388         error_str += channel;
389         error_str += " is not in message map";
390
391         #ifdef _WIN32
392             std::cout << error_str << std::endl;
393         #endif /* _WIN32 */
394
395         throw std::runtime_error(error_str);
396     }
397
398     // 2. check if channel is empty (if so, throw error)
399     if (this->message_map[channel].empty()) {
400         std::string error_str = "ERROR MessageHub::receiveMessage() channel ";
401         error_str += channel;
402         error_str += " is empty";
403
404         #ifdef _WIN32
405             std::cout << error_str << std::endl;
406         #endif /* _WIN32 */
407
408         throw std::runtime_error(error_str);
409     }
410 }

```



```

411 // 3. pop message
412 this->message_map[channel].pop_front();
413
414 return;
415 } /* popMessage() */

```

4.9.3.7 printState()

```

void MessageHub::printState (
    void )

```

Method for printing message hub state information (mostly for troubleshooting message deadlocks).

```

203 {
204     std::cout << "\n\n    **** MESSAGE HUB STATE ****    \n" << std::endl;
205
206     std::map<std::string, std::list<Message>::iterator> channel_iterator;
207
208     for (
209         channel_iterator = this->message_map.begin();
210         channel_iterator != this->message_map.end();
211         channel_iterator++
212     ) {
213         std::string channel = channel_iterator->first;
214         std::list<Message> message_queue = channel_iterator->second;
215
216         std::cout << "-----" << std::endl;
217         std::cout << "\tCHANNEL: " << channel << std::endl;
218         std::cout << "\tMESSAGE QUEUE LENGTH: " << message_queue.size() << std::endl;
219         std::cout << std::endl;
220
221         std::list<Message>::iterator message_queue_iterator;
222
223         for (
224             message_queue_iterator = message_queue.begin();
225             message_queue_iterator != message_queue.end();
226             message_queue_iterator++
227         ) {
228             std::cout << "\tSUBJECT: " << (*message_queue_iterator).subject <<
229                 std::endl;
230         }
231
232         std::cout << std::endl;
233     }
234
235     std::cout << std::endl;
236
237     return;
238 } /* printState() */

```

4.9.3.8 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.

```

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

```

4.9.3.9 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() */

```

4.9.3.10 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.
----------------	-------------------------

```
256 {
257     // 1. check if channel is in map (if not, throw error)
258     std::string channel = message.channel;
259
260     if (this->message_map.count(channel) <= 0) {
261         std::string error_str = "ERROR MessageHub::sendMessage() channel ";
262         error_str += channel;
263         error_str += " is not in message map";
264
265         #ifdef _WIN32
266             std::cout << error_str << std::endl;
267         #endif /* _WIN32 */
268
269         throw std::runtime_error(error_str);
270     }
271
272     // 2. send message to message map
273     this->message_map[channel].push_back(message);
274
275     return;
276 } /* 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:

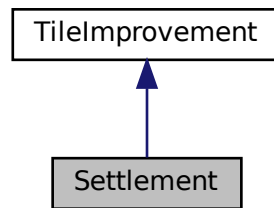
- header/ESC_core/[MessageHub.h](#)
- source/ESC_core/[MessageHub.cpp](#)

4.10 Settlement Class Reference

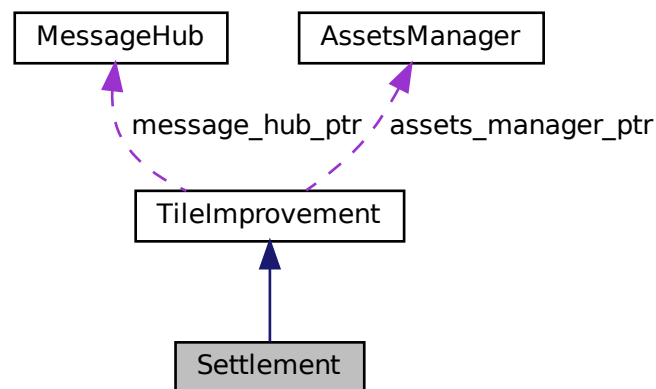
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for Settlement:



Collaboration diagram for Settlement:



Public Member Functions

- `Settlement` (double, double, int, 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

- bool [draw_coin](#)
Boolean indicating whether or not to draw credits earned coin.
- 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).
- sf::Sprite [coin_sprite](#)
A coin sprite (for credits earned animation).

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__setUpCoinSprite](#) (void)
Helper method to set up and place coin sprite.
- 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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.

```

241 :
242 TileImprovement (
243     position_x,
244     position_y,
245     tile_resource,
246     event_ptr,
247     render_window_ptr,
248     assets_manager_ptr,
249     message_hub_ptr
250 )
251 {
252     // 1. set attributes
253
254     // 1.1. private
255     //...
256
257     // 1.2. public
258     this->tile_improvement_type = TileImprovementType :: SETTLEMENT;
259
260     this->draw_coin = false;
261
262     this->smoke_da = SECONDS_PER_FRAME / 4;
263     this->smoke_dx = 5 * SECONDS_PER_FRAME;
264     this->smoke_dy = -10 * SECONDS_PER_FRAME;
265     this->smoke_prob = 3 * SECONDS_PER_FRAME;
266
267     this->smoke_sprite_list = {};
268
269     this->tile_improvement_string = "SETTLEMENT";
270
271     this->__setUpTileImprovementSpriteStatic();
272     this->__setUpCoinSprite();
273
274     this->message_hub_ptr->addChannel (SETTLEMENT_CHANNEL);
275
276     std::cout << "Settlement constructed at " << this << std::endl;
277
278     return;
279 } /* Settlement() */

```

4.10.2.2 ~Settlement()

```

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

```

Destructor for the [Settlement](#) class.

```

502 {
503     std::cout << "Settlement at " << this << " destroyed" << std::endl;
504
505     return;
506 } /* ~Settlement() */

```

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
131 {
132     if (this->just_built) {
133         return;
134     }
135
136     switch (this->event_ptr->key.code) {
137         //...
138
139         default: {
140             // do nothing!
141
142             break;
143         }
144     }
145
146     return;
147 } /* __handleKeyPressEvents() */
```

4.10.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
163 {
164     if (this->just_built) {
165         return;
166     }
167
168     switch (this->event_ptr->mouseButton.button) {
169         case (sf::Mouse::Left): {
170             //...
171
172             break;
173         }
174
175         case (sf::Mouse::Right): {
176             //...
177
178             break;
179         }
180     }
181
182     default: {
183         // do nothing!
184
185         break;
186     }
187 }
188
189 return;
191 } /* __handleMouseButtonEvents() */
```

4.10.3.3 __setUpCoinSprite()

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

Helper method to set up and place coin sprite.

```
103 {
104     this->coin_sprite.setTexture(
105         *(this->assets_manager_ptr->getTexture("coin"))
106     );
107
108     this->coin_sprite.setOrigin(
109         this->coin_sprite.getLocalBounds().width / 2,
110         this->coin_sprite.getLocalBounds().height / 2
111     );
112
113     this->coin_sprite.setPosition(this->position_x, this->position_y);
114
115     return;
116 } /* __setUpCoinSprite() */
```

4.10.3.4 __setUpTileImprovementSpriteStatic()

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

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

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

4.10.3.5 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](#).

```
409 {
410     // 1. if just built, call base method and return
411     if (this->just_built) {
412         TileImprovement::draw();
413
414         return;
415     }
416 }
```



```

417 // 2. draw static sprite and chimney smoke effects
418 this->render_window_ptr->draw(this->tile_improvement_sprite_static);
419
420 std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
421
422 double alpha = 255;
423
424 while (iter != this->smoke_sprite_list.end()) {
425     this->render_window_ptr->draw(*iter);
426
427     alpha = (*iter).getColor().a;
428
429     alpha -= this->smoke_da;
430
431     if (alpha <= 0) {
432         iter = this->smoke_sprite_list.erase(iter);
433         continue;
434     }
435
436     (*iter).setColor(sf::Color(255, 255, 255, alpha));
437
438     (*iter).move(
439         this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
440         this->smoke_dy
441     );
442
443     (*iter).rotate((((double)rand() / RAND_MAX)));
444
445     iter++;
446 }
447
448 if ((double)rand() / RAND_MAX < smoke_prob) {
449     this->smoke_sprite_list.push_back(
450         sf::Sprite(*this->assets_manager_ptr->getTexture("emissions")))
451     );
452
453     this->smoke_sprite_list.back().setOrigin(
454         this->smoke_sprite_list.back().getLocalBounds().width / 2,
455         this->smoke_sprite_list.back().getLocalBounds().height / 2
456     );
457
458     this->smoke_sprite_list.back().setPosition(
459         this->position_x + 9 + 4 * (((double)rand() / RAND_MAX) - 2,
460         this->position_y - 33
461     );
462     );
463 }
464
465
466 // 4. draw coin
467 if (this->draw_coin) {
468     double alpha = this->coin_sprite.getColor().a;
469
470     alpha -= this->smoke_da;
471
472     if (alpha <= 0) {
473         this->coin_sprite.setColor(sf::Color(255, 255, 255, 255));
474         this->coin_sprite.setPosition(this->position_x, this->position_y);
475         this->draw_coin = false;
476     }
477
478     this->coin_sprite.move(0, this->smoke_dy);
479     this->coin_sprite.setColor(sf::Color(255, 255, 255, alpha));
480
481     this->render_window_ptr->draw(this->coin_sprite);
482 }
483
484 this->frame++;
485 return;
486 }
487 } /* draw() */

```

4.10.3.6 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

321 {
322     //          32 char x 17 line console "-----\n";
323     std::string options_substring          = "    **** SETTLEMENT OPTIONS **** \n";
324     options_substring                     += " \n";
325     options_substring                     += " \n";
326     options_substring                     += " \n";
327     options_substring                     += " \n";
328     options_substring                     += " \n";
329     options_substring                     += " \n";
330     options_substring                     += " \n";
331
332     return options_substring;
333 } /* getTileOptionsSubstring() */

```

4.10.3.7 processEvent()

```

void Settlement::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

348 {
349     TileImprovement :: processEvent();
350
351     if (this->event_ptr->type == sf::Event::KeyPressed) {
352         this->__handleKeyPressEvents();
353     }
354
355     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
356         this->__handleMouseButtonEvents();
357     }
358
359     return;
360 } /* processEvent() */

```

4.10.3.8 processMessage()

```

void Settlement::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

375 {
376     TileImprovement :: processMessage();
377
378     if (not this->message_hub_ptr->isEmpty(SETTLEMENT_CHANNEL)) {
379         Message settlement_message = this->message_hub_ptr->receiveMessage(
380             SETTLEMENT_CHANNEL
381         );
382
383         if (settlement_message.subject == "credits earned") {
384             this->draw_coin = true;
385             this->assets_manager_ptr->getSound("coin ring")->play();
386
387             std::cout << "Credits earned message received by " << this << std::endl;
388             this->message_hub_ptr->popMessage(SETTLEMENT_CHANNEL);
389         }
390     }
391
392     return;
393 } /* processMessage() */

```

4.10.3.9 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](#).

```
296 {
297     TileImprovement :: setIsSelected(is_selected);
298
299     if (this->is_selected) {
300         this->assets_manager_ptr->getSound("people and children")->play();
301     }
302
303     return;
304 } /* setIsSelected() */
```

4.10.4 Member Data Documentation

4.10.4.1 coin_sprite

```
sf::Sprite Settlement::coin_sprite
```

A coin sprite (for credits earned animation).

4.10.4.2 draw_coin

```
bool Settlement::draw_coin
```

Boolean indicating whether or not to draw credits earned coin.

4.10.4.3 smoke_da

```
double Settlement::smoke_da
```

The per frame delta in smoke particle alpha value.

4.10.4.4 smoke_dx

```
double Settlement::smoke_dx
```

The per frame delta in smoke particle x position.

4.10.4.5 smoke_dy

```
double Settlement::smoke_dy
```

The per frame delta in smoke particle y position.

4.10.4.6 smoke_prob

```
double Settlement::smoke_prob
```

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

4.10.4.7 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:

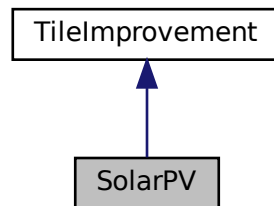
- header/[Settlement.h](#)
- source/[Settlement.cpp](#)

4.11 SolarPV Class Reference

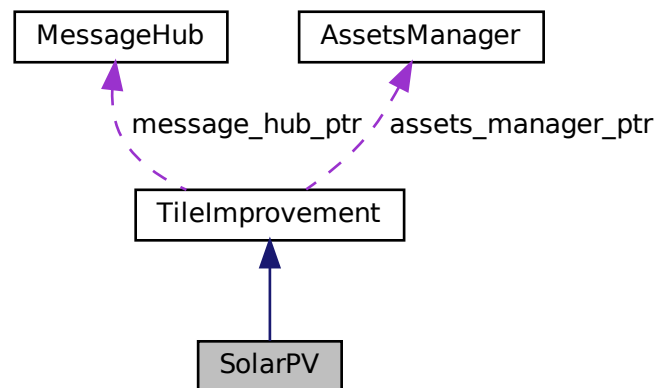
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for SolarPV:



Collaboration diagram for SolarPV:



Public Member Functions

- [SolarPV](#) (double, double, int, 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 [setIsSelected](#) (bool)
Method to set the is selected attribute.
- void [advanceTurn](#) (void)

- *Method to handle turn advance.*
- void [update](#) (void)
- *Method to trigger production and dispatchable updates.*
- 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 [dispatch_MWh](#)
- *The current dispatch [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).*
- double [max_daily_production_MWh](#)
- *The maximum daily production [MWh] of the solar PV array.*
- std::vector< double > [capacity_factor_vec](#)
- *A vector of daily capacity factors for the current month.*
- std::vector< double > [production_vec_MWh](#)
- *A vector of daily production [MWh] for the current month.*
- std::vector< double > [dispatch_vec_MWh](#)
- *A vector of daily dispatch [MWh] for the current month.*

Private Member Functions

- void [__setUpTileImprovementSpriteStatic](#) (void)
- *Helper method to set up tile improvement sprite (static).*
- void [__drawProductionMenu](#) (void)
- *Helper method to draw production menu assets.*
- void [__upgradePowerCapacity](#) (void)
- *Helper method to upgrade power capacity.*
- void [__computeProductionCosts](#) (void)
- *Helper method to compute production costs (O&M) based on current production level.*
- void [__breakdown](#) (void)
- *Helper method to trigger an equipment breakdown.*
- void [__computeCapacityFactors](#) (void)
- *Helper method to compute capacity factors.*
- void [__computeProduction](#) (void)
- *Helper method to compute production values.*
- void [__computeDispatch](#) (void)
- *Helper method to compute dispatch values.*

- 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.
- void [__sendImprovementStateMessage](#) (void)
Helper method to format and sent improvement state message.

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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.

```
712 :
713 TileImprovement (
714     position_x,
715     position_y,
716     tile_resource,
```

```

717     event_ptr,
718     render_window_ptr,
719     assets_manager_ptr,
720     message_hub_ptr
721 )
722 {
723     // 1. set attributes
724
725     // 1.1. private
726     //...
727
728     // 1.2. public
729     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
730
731     this->is_running = false;
732
733     this->health = 100;
734
735     this->capacity_kW = 100;
736     this->upgrade_level = 1;
737
738     this->storage_kWh = 0;
739     this->storage_level = 0;
740
741     this->production_MWh = 0;
742     this->dispatch_MWh = 0;
743     this->dispatchable_MWh = 0;
744
745     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
746
747     this->capacity_factor_vec.resize(30, 0);
748     this->production_vec_MWh.resize(30, 0);
749     this->dispatch_vec_MWh.resize(30, 0);
750
751     this->tile_improvement_string = "SOLAR PV ARRAY";
752
753     this->__setUpTileImprovementSpriteStatic();
754     this->__computeCapacityFactors();
755     this->update();
756
757     std::cout << "SolarPV constructed at " << this << std::endl;
758
759     return;
760 } /* SolarPV() */

```

4.11.2.2 ~SolarPV()

```

SolarPV::~SolarPV (
    void ) [virtual]

```

Destructor for the [SolarPV](#) class.

```

1092 {
1093     std::cout << "SolarPV at " << this << " destroyed" << std::endl;
1094
1095     return;
1096 } /* ~SolarPV() */

```

4.11.3 Member Function Documentation

4.11.3.1 __breakdown()

```

void SolarPV::__breakdown (
    void ) [private]

```

Helper method to trigger an equipment breakdown.


```

233 {
234     TileImprovement :: __breakdown();
235
236     this->production_MWh = 0;
237     this->dispatch_MWh = 0;
238     this->dispatchable_MWh = 0;
239     this->operation_maintenance_cost = 0;
240
241     return;
242 } /* __breakdown() */

```

4.11.3.2 __computeCapacityFactors()

```

void SolarPV::__computeCapacityFactors (
    void ) [private]

```

Helper method to compute capacity factors.

```

257 {
258     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
259     std::default_random_engine generator(seed);
260
261     double mean =
262         this->tile_resource_scalar * MEAN_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
263
264     double stdev = STDEV_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
265
266     if (this->tile_resource_scalar > 1) {
267         stdev /= this->tile_resource_scalar;
268     }
269
270     std::normal_distribution<double> normal_dist(mean, stdev);
271
272     double capacity_factor = 0;
273
274     for (int i = 0; i < 30; i++) {
275         capacity_factor = normal_dist(generator);
276
277         if (capacity_factor < 0) {
278             capacity_factor = 0;
279         }
280
281         this->capacity_factor_vec[i] = capacity_factor;
282     }
283
284     return;
285 } /* __computeCapacityFactors() */

```

4.11.3.3 __computeDispatch()

```

void SolarPV::__computeDispatch (
    void ) [private]

```

Helper method to compute dispatch values.

```

328 {
329     double stored_energy_MWh = 0;
330     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
331
332     double demand_MWh = 0;
333     double production_MWh = 0;
334     double dispatchable_MWh = 0;
335     double difference_MWh = 0;
336
337     double room_MWh = 0;
338
339     for (int i = 0; i < 30; i++) {
340         demand_MWh = this->demand_vec_MWh[i];
341         production_MWh = this->production_vec_MWh[i];
342
343         if (production_MWh <= demand_MWh) {

```

```

344         this->dispatch_vec_MWh[i] = production_MWh;
345         dispatchable_MWh += this->dispatch_vec_MWh[i];
346
347         difference_MWh = demand_MWh - production_MWh;
348
349         if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
350             if (difference_MWh > stored_energy_MWh) {
351                 this->dispatch_vec_MWh[i] += stored_energy_MWh;
352                 dispatchable_MWh += stored_energy_MWh;
353                 stored_energy_MWh = 0;
354             }
355
356             else {
357                 this->dispatch_vec_MWh[i] += difference_MWh;
358                 dispatchable_MWh += difference_MWh;
359                 stored_energy_MWh -= difference_MWh;
360             }
361         }
362     }
363
364     else {
365         this->dispatch_vec_MWh[i] = demand_MWh;
366         dispatchable_MWh += this->dispatch_vec_MWh[i];
367
368         difference_MWh = production_MWh - demand_MWh;
369
370         if (
371             (storage_capacity_MWh > 0) and
372             (stored_energy_MWh < storage_capacity_MWh)
373         ) {
374             room_MWh = storage_capacity_MWh - stored_energy_MWh;
375
376             if (difference_MWh > room_MWh) {
377                 stored_energy_MWh += room_MWh;
378             }
379
380             else {
381                 stored_energy_MWh += difference_MWh;
382             }
383         }
384     }
385 }
386
387 this->dispatchable_MWh = round(dispatchable_MWh);
388
389 if (this->dispatch_MWh > this->dispatchable_MWh) {
390     this->dispatch_MWh = this->dispatchable_MWh;
391 }
392
393 return;
394 } /* __computeDispatch() */

```

4.11.3.4 __computeProduction()

```

void SolarPV::__computeProduction (
    void ) [private]

```

Helper method to compute production values.

```

300 {
301     double production_MWh = 0;
302
303     for (int i = 0; i < 30; i++) {
304         this->production_vec_MWh[i] =
305             this->max_daily_production_MWh * this->capacity_factor_vec[i];
306
307         production_MWh += this->production_vec_MWh[i];
308     }
309
310     this->production_MWh = round(production_MWh);
311
312     return;
313 } /* __computeProduction() */

```

4.11.3.5 __computeProductionCosts()

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

Helper method to compute production costs (O&M) based on current production level.

```
212 {
213     double operation_maintenance_cost =
214         (this->production_MWh * SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
215     this->operation_maintenance_cost = round(operation_maintenance_cost);
216
217     return;
218 } /* __computeProductionCosts() */
```

4.11.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
103 {
104     // 1. draw static sprite
105     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
106     this->tile_improvement_sprite_static.setPosition(400 - 138, 400 + 16);
107
108     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
109     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
110
111     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();
112     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
113
114     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
115
116     this->tile_improvement_sprite_static.setPosition(initial_position);
117     this->tile_improvement_sprite_static.setColor(initial_colour);
118     this->tile_improvement_sprite_static.setScale(initial_scale);
119
120     // 2. draw production text
121     std::string production_string = "[W]: INCREASE DISPATCH\n";
122     production_string             += "[S]: DECREASE DISPATCH\n";
123     production_string             += "\n";
124
125     production_string             += "DISPATCH: ";
126     production_string             += std::to_string(this->dispatch_MWh);
127     production_string             += " MWh (MAX ";
128     production_string             += std::to_string(this->dispatchable_MWh);
129     production_string             += ")\n";
130
131     production_string             += "O&M COST: ";
132     production_string             += std::to_string(this->operation_maintenance_cost);
133     production_string             += " K\n";
134
135     sf::Text production_text(
136         production_string,
137         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
138         16
139     );
140
141     production_text.setOrigin(production_text.getLocalBounds().width / 2, 0);
142     production_text.setFillColor(MONOCROME_TEXT_GREEN);
143
144     production_text.setPosition(400 + 30, 400 - 45);
145
146     this->render_window_ptr->draw(production_text);
147
148     return;
149 } /* __drawProductionMenu() */
```

4.11.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
535 {
536     // 1. draw power capacity upgrade sprite
537     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
538     this->tile_improvement_sprite_static.setPosition(400 - 100, 400 - 32);
539
540     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
541     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
542
543     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();
544     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
545
546     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
547
548     this->tile_improvement_sprite_static.setPosition(initial_position);
549     this->tile_improvement_sprite_static.setColor(initial_colour);
550     this->tile_improvement_sprite_static.setScale(initial_scale);
551
552     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
553
554
555     // 2. draw power capacity upgrade text
556     // 16 char line = "                                \n"
557     std::string power_upgrade_string = "POWER CAPACITY \n";
558     power_upgrade_string += "                                \n";
559
560     power_upgrade_string += "CAPACITY: ";
561     power_upgrade_string += std::to_string(this->capacity_kW);
562     power_upgrade_string += " kW\n";
563
564     power_upgrade_string += "LEVEL: ";
565     power_upgrade_string += std::to_string(this->upgrade_level);
566     power_upgrade_string += "\n\n";
567
568     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
569         power_upgrade_string += "[W]: + 100 kW (";
570         power_upgrade_string += std::to_string(SOLAR_PV_BUILD_COST);
571         power_upgrade_string += " K)\n";
572     }
573
574     else {
575         power_upgrade_string += " * MAX LEVEL * \n";
576     }
577
578     sf::Text power_upgrade_text = sf::Text(
579         power_upgrade_string,
580         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
581         16
582     );
583
584     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
585     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
586     power_upgrade_text.setFillColors(MONOCHROME_TEXT_GREEN);
587
588     this->render_window_ptr->draw(power_upgrade_text);
589
590
591     // 3. draw energy capacity (storage) upgrade sprite
592     this->render_window_ptr->draw(this->storage_upgrade_sprite);
593     this->render_window_ptr->draw(this->upgrade_plus_sprite);
594
595
596     // 4. draw energy capacity (storage) upgrade text
597     // 16 char line = "                                \n"
598     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
599     energy_upgrade_string += "                                \n";
600
601     energy_upgrade_string += "CAPACITY: ";
602     energy_upgrade_string += std::to_string(this->storage_level * 200);
603     energy_upgrade_string += " kWh\n";
604
605     energy_upgrade_string += "LEVEL: ";
606     energy_upgrade_string += std::to_string(this->storage_level);
607     energy_upgrade_string += "\n\n";
608
609     if (this->storage_level < MAX_STORAGE_LEVELS) {
610         energy_upgrade_string += "[D]: + 200 kWh (";
611         energy_upgrade_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
612         energy_upgrade_string += " K)\n";
613     }
```

```

613     }
614
615     else {
616         energy_upgrade_string += " * MAX LEVEL * \n";
617     }
618
619     sf::Text energy_upgrade_text = sf::Text(
620         energy_upgrade_string,
621         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
622         16
623     );
624
625     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
626     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
627     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
628
629     this->render_window_ptr->draw(energy_upgrade_text);
630
631     return;
632 } /* __drawUpgradeOptions() */

```

4.11.3.8 __handleKeyPressEvents()

```

void SolarPV::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

409 {
410     if (this->just_built) {
411         return;
412     }
413
414     switch (this->event_ptr->key.code) {
415         case (sf::Keyboard::U): {
416             this->__openUpgradeMenu();
417             break;
418         }
419
420         case (sf::Keyboard::W): {
421             if (this->production_menu_open) {
422                 this->dispatch_MWh++;
423
424                 if (this->dispatch_MWh > this->dispatchable_MWh) {
425                     this->dispatch_MWh = 0;
426                 }
427
428                 this->__computeProductionCosts();
429                 this->assets_manager_ptr->getSound("interface click")->play();
430             }
431
432             else if (this->upgrade_menu_open) {
433                 this->__upgradePowerCapacity();
434             }
435
436             break;
437         }
438
439         case (sf::Keyboard::S): {
440             if (this->production_menu_open) {
441                 this->dispatch_MWh--;
442
443                 if (this->dispatch_MWh < 0) {
444                     this->dispatch_MWh = this->dispatchable_MWh;
445                 }
446
447                 this->__computeProductionCosts();
448                 this->assets_manager_ptr->getSound("interface click")->play();
449             }
450
451             break;
452         }
453
454         case (sf::Keyboard::D): {
455             if (this->upgrade_menu_open) {

```

```

460         this->__upgradeStorageCapacity();
461         this->__computeProduction();
462         this->__computeDispatch();
463     }
464
465     break;
466 }
467
468
469     default: {
470         // do nothing!
471
472         break;
473     }
474 }
475
476 return;
477 } /* __handleKeyPressEvents() */

```

4.11.3.9 __handleMouseButtonEvents()

```

void SolarPV::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

492 {
493     if (this->just_built) {
494         return;
495     }
496
497     switch (this->event_ptr->mouseButton.button) {
498         case (sf::Mouse::Left): {
499             //...
500
501             break;
502         }
503
504         case (sf::Mouse::Right): {
505             //...
506
507             break;
508         }
509
510         default: {
511             // do nothing!
512
513             break;
514         }
515     }
516
517     return;
518 }
519
520 } /* __handleMouseButtonEvents() */

```

4.11.3.10 __sendImprovementStateMessage()

```

void SolarPV::__sendImprovementStateMessage (
    void ) [private]

```

Helper method to format and sent improvement state message.

```

647 {
648     Message improvement_state_message;
649
650     improvement_state_message.channel = GAME_CHANNEL;
651     improvement_state_message.subject = "improvement state";
652
653     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
654     improvement_state_message.int_payload["operation_maintenance_cost"] =

```

```

655         this->operation_maintenance_cost;
656
657     this->message_hub_ptr->sendMessage(improvement_state_message);
658
659     std::cout << "Improvement state message sent by " << this << std::endl;
660
661     return;
662 } /* __sendImprovementStateMessage() */

```

4.11.3.11 __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() */

```

4.11.3.12 __upgradePowerCapacity()

```

void SolarPV::__upgradePowerCapacity (
    void ) [private]

```

Helper method to upgrade power capacity.

```

164 {
165     if (this->credits < SOLAR_PV_BUILD_COST) {
166         std::cout << "Cannot upgrade solar PV: insufficient credits (need "
167             << SOLAR_PV_BUILD_COST << " K)" << std::endl;
168
169         this->__sendInsufficientCreditsMessage();
170         return;
171     }
172
173     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
174         return;
175     }
176
177     this->health = 100;
178
179     this->capacity_kW += 100;
180     this->upgrade_level++;
181
182     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
183
184     this->__computeProduction();
185     this->__computeDispatch();
186
187     this->just_upgraded = true;
188
189     this->assets_manager_ptr->getSound("upgrade")->play();

```

```

190
191     this->__sendCreditsSpentMessage(SOLAR_PV_BUILD_COST);
192     this->__sendTileStateRequest();
193     this->__sendGameStateRequest();
194
195     return;
196 } /* __upgradePowerCapacity() */

```

4.11.3.13 advanceTurn()

```

void SolarPV::advanceTurn (
    void ) [virtual]

```

Method to handle turn advance.

Reimplemented from [TileImprovement](#).

```

865 {
866     // 1. update
867     this->__computeCapacityFactors();
868     this->update();
869
870     // 2. send improvement state message
871     this->__sendImprovementStateMessage();
872
873     // 3. handle start/stop
874     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
875         this->is_running = true;
876     }
877
878     else if (this->is_running and (this->dispatch_MWh <= 0)) {
879         this->is_running = false;
880     }
881
882     // 4. handle equipment health
883     if (this->is_running) {
884         this->health--;
885
886         if (this->health <= 0) {
887             this->__breakdown();
888         }
889     }
890
891     // 5. send tile state request (if selected)
892     if (this->is_selected) {
893         this->__sendTileStateRequest();
894     }
895
896     return;
897 } /* advanceTurn() */

```

4.11.3.14 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](#).

```

986 {
987     // 1. if just built, call base method and return
988     if (this->just_built) {
989         TileImprovement::draw();
990
991         return;
992     }
993
994

```



```

995 // 2. handle upgrade effects
996 if (this->just_upgraded) {
997     this->tile_improvement_sprite_static.setColor(
998         sf::Color(
999             255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1000             255,
1001             255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1002             255
1003         )
1004     );
1005
1006     this->tile_improvement_sprite_static.setScale(
1007         sf::Vector2f(
1008             1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1009             1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1010         )
1011     );
1012
1013     this->upgrade_frame++;
1014 }
1015
1016 if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1017     this->tile_improvement_sprite_static.setColor(
1018         sf::Color(255,255,255,255)
1019     );
1020
1021     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
1022
1023     this->just_upgraded = false;
1024     this->upgrade_frame = 0;
1025 }
1026
1027 // 3. draw static sprite
1028 this->render_window_ptr->draw(this->tile_improvement_sprite_static);
1029
1030 // 4. draw storage upgrades
1031 for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1032     this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1033 }
1034
1035 // 5. handle dispatch illustration
1036 if (this->dispatch_MWh > 0) {
1037     this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1038     this->__drawDispatch();
1039 }
1040
1041 // 6. draw production menu
1042 if (this->production_menu_open) {
1043     this->render_window_ptr->draw(this->production_menu_backing);
1044     this->render_window_ptr->draw(this->production_menu_backing_text);
1045
1046     this->__drawProductionMenu();
1047 }
1048
1049 // 7. draw upgrade menu
1050 if (this->upgrade_menu_open) {
1051     this->render_window_ptr->draw(this->upgrade_menu_backing);
1052     this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1053
1054     this->__drawUpgradeOptions();
1055 }
1056
1057 // 10. handle broken effects
1058 if (this->is_broken) {
1059     this->tile_improvement_sprite_static.setColor(
1060         sf::Color(
1061             255,
1062             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1063             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1064             255
1065         )
1066     );
1067 }
1068
1069 this->frame++;
1070 return;
1071 } /* draw() */

```

4.11.3.15 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```
777 {
778     //          32 char x 17 line console "-----\n";
779     std::string options_substring = "CAPACITY: ";
780     options_substring += std::to_string(this->capacity_kW);
781     options_substring += " kW (level ";
782     options_substring += std::to_string(this->upgrade_level);
783     options_substring += ")\n";
784
785     options_substring += "PRODUCTION: ";
786     options_substring += std::to_string(this->production_MWh);
787     options_substring += " MWh\n";
788
789     options_substring += "DISPATCHABLE: ";
790     options_substring += std::to_string(this->dispatchable_MWh);
791     options_substring += " MWh\n";
792
793     options_substring += "HEALTH: ";
794     options_substring += std::to_string(this->health);
795     options_substring += "/100";
796
797     if (this->health <= 0) {
798         options_substring += " ** BROKEN! **\n";
799     }
800
801     else {
802         options_substring += "\n";
803     }
804
805     options_substring += "
806     options_substring += "      **** SOLAR PV OPTIONS ****
807     options_substring += "
808
809     options_substring += "      [E]: ";
810
811     if (this->is_broken) {
812         options_substring += "*** BROKEN! ***\n";
813     }
814
815     else {
816         options_substring += "OPEN PRODUCTION MENU\n";
817     }
818
819     options_substring += "      [U]: OPEN UPGRADE MENU
820     options_substring += "HOLD [P]: SCRAP ("
821     options_substring += std::to_string(SCRAP_COST);
822     options_substring += " K)";
823
824     return options_substring;
825 } /* getTileOptionsSubstring() */
```

4.11.3.16 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```

937 {
938     TileImprovement :: processEvent ();
939
940     if (this->event_ptr->type == sf::Event::KeyPressed) {
941         this->__handleKeyPressEvents ();
942     }
943
944     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
945         this->__handleMouseButtonEvents ();
946     }
947
948     return;
949 } /* processEvent () */

```

4.11.3.17 processMessage()

```

void SolarPV::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

964 {
965     TileImprovement :: processMessage ();
966
967     //...
968
969     return;
970 } /* processMessage () */

```

4.11.3.18 setIsSelected()

```

void SolarPV::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](#).

```

842 {
843     TileImprovement :: setIsSelected(is_selected);
844
845     if (this->is_running and this->is_selected) {
846         this->assets_manager_ptr->getSound("solar hum")->play();
847     }
848
849     return;
850 } /* setIsSelected() */

```

4.11.3.19 update()

```

void SolarPV::update (
    void ) [virtual]

```

Method to trigger production and dispatchable updates.

Reimplemented from [TileImprovement](#).

```

912 {
913     this->__computeProduction();
914     this->__computeProductionCosts();
915     this->__computeDispatch();
916
917     if (this->is_selected) {
918         this->__sendTileStateRequest();
919     }
920
921     return;
922 } /* update() */

```

4.11.4 Member Data Documentation

4.11.4.1 capacity_factor_vec

```
std::vector<double> SolarPV::capacity_factor_vec
```

A vector of daily capacity factors for the current month.

4.11.4.2 capacity_kW

```
int SolarPV::capacity_kW
```

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

4.11.4.3 dispatch_MWh

```
int SolarPV::dispatch_MWh
```

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

4.11.4.4 dispatch_vec_MWh

```
std::vector<double> SolarPV::dispatch_vec_MWh
```

A vector of daily dispatch [MWh] for the current month.

4.11.4.5 dispatchable_MWh

```
int SolarPV::dispatchable_MWh
```

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

4.11.4.6 max_daily_production_MWh

```
double SolarPV::max_daily_production_MWh
```

The maximum daily production [MWh] of the solar PV array.

4.11.4.7 production_MWh

```
int SolarPV::production_MWh
```

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

4.11.4.8 production_vec_MWh

```
std::vector<double> SolarPV::production_vec_MWh
```

A vector of daily production [MWh] for the current month.

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

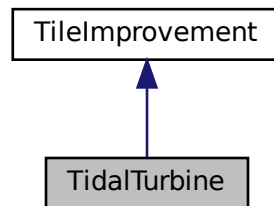
- header/[SolarPV.h](#)
- source/[SolarPV.cpp](#)

4.12 TidalTurbine Class Reference

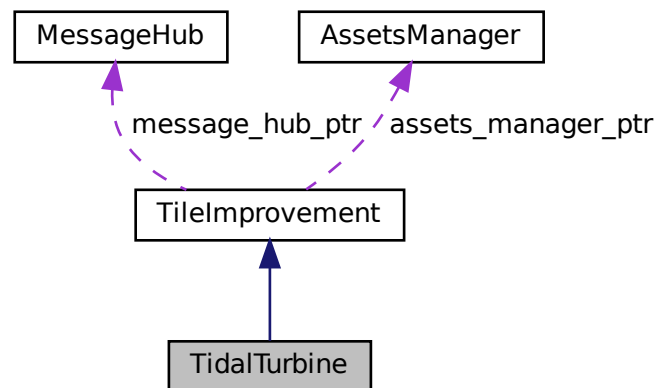
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for TidalTurbine:



Collaboration diagram for TidalTurbine:



Public Member Functions

- [TidalTurbine](#) (double, double, int, 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 [setIsSelected](#) (bool)
Method to set the is selected attribute.
- void [advanceTurn](#) (void)

- *Method to handle turn advance.*
- void [update](#) (void)
- *Method to trigger production and dispatchable updates.*
- 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 [dispatch_MWh](#)
- *The current dispatch [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).*
- double [max_daily_production_MWh](#)
- *The maximum daily production [MWh] of the solar PV array.*
- double [rotor_drotation](#)
- *The rotation rate of the rotor.*
- double [bobbing_y](#)
- *The bobbing extent of the tidal turbine.*
- std::vector< double > [capacity_factor_vec](#)
- *A vector of daily capacity factors for the current month.*
- std::vector< double > [production_vec_MWh](#)
- *A vector of daily production [MWh] for the current month.*
- std::vector< double > [dispatch_vec_MWh](#)
- *A vector of daily dispatch [MWh] for the current month.*

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
- *Helper method to set up tile improvement sprite (static).*
- void [__drawProductionMenu](#) (void)
- *Helper method to draw production menu assets.*
- void [__upgradePowerCapacity](#) (void)
- *Helper method to upgrade power capacity.*
- void [__computeProductionCosts](#) (void)
- *Helper method to compute production costs (O&M) based on current production level.*
- void [__breakdown](#) (void)
- *Helper method to trigger an equipment breakdown.*
- void [__computeCapacityFactors](#) (void)
- *Helper method to compute capacity factors.*

- void [__computeProduction](#) (void)
Helper method to compute production values.
- void [__computeDispatch](#) (void)
Helper method to compute dispatch values.
- 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.
- void [__sendImprovementStateMessage](#) (void)
Helper method to format and sent improvement state message.

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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.


```

714 :
715 TileImprovement (
716     position_x,
717     position_y,
718     tile_resource,
719     event_ptr,
720     render_window_ptr,
721     assets_manager_ptr,
722     message_hub_ptr
723 )
724 {
725     // 1. set attributes
726
727     // 1.1. private
728     //...
729
730     // 1.2. public
731     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
732
733     this->is_running = false;
734
735     this->health = 100;
736
737     this->capacity_kW = 100;
738     this->upgrade_level = 1;
739
740     this->storage_kWh = 0;
741     this->storage_level = 0;
742
743     this->production_MWh = 0;
744     this->dispatch_MWh = 0;
745     this->dispatchable_MWh = 0;
746
747     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
748
749     this->rotor_drotation = 64 * SECONDS_PER_FRAME;
750     this->bobbing_y = 4;
751
752     this->capacity_factor_vec.resize(30, 0);
753     this->production_vec_MWh.resize(30, 0);
754     this->dispatch_vec_MWh.resize(30, 0);
755
756     this->tile_improvement_string = "TIDAL TURBINE";
757
758     this->__setUpTileImprovementSpriteAnimated();
759     this->__computeCapacityFactors();
760     this->update();
761
762     std::cout << "TidalTurbine constructed at " << this << std::endl;
763
764     return;
765 } /* TidalTurbine() */

```

4.12.2.2 ~TidalTurbine()

```

TidalTurbine::~TidalTurbine (
    void ) [virtual]

```

Destructor for the [TidalTurbine](#) class.

```

1123 {
1124     std::cout << "TidalTurbine at " << this << " destroyed" << std::endl;
1125
1126     return;
1127 } /* ~TidalTurbine() */

```

4.12.3 Member Function Documentation

4.12.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
250 {
251     TileImprovement :: __breakdown();
252
253     this->production_MWh = 0;
254     this->dispatch_MWh = 0;
255     this->dispatchable_MWh = 0;
256     this->operation_maintenance_cost = 0;
257
258     return;
259 } /* __breakdown() */
```

4.12.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
274 {
275     for (int i = 0; i < 30; i++) {
276         this->capacity_factor_vec[i] =
277             this->tile_resource_scalar * DAILY_TIDAL_CAPACITY_FACTOR;
278     }
279
280     return;
281 } /* __computeCapacityFactors() */
```

4.12.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
324 {
325     double stored_energy_MWh = 0;
326     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
327
328     double demand_MWh = 0;
329     double production_MWh = 0;
330     double dispatchable_MWh = 0;
331     double difference_MWh = 0;
332
333     double room_MWh = 0;
334
335     for (int i = 0; i < 30; i++) {
336         demand_MWh = this->demand_vec_MWh[i];
337         production_MWh = this->production_vec_MWh[i];
338
339         if (production_MWh <= demand_MWh) {
340             this->dispatch_vec_MWh[i] = production_MWh;
341             dispatchable_MWh += this->dispatch_vec_MWh[i];
342
343             difference_MWh = demand_MWh - production_MWh;
344
345             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
346                 if (difference_MWh > stored_energy_MWh) {
347                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
348                     dispatchable_MWh += stored_energy_MWh;
349                     stored_energy_MWh = 0;
350                 }
351             }
352         }
353     }
354 }
```

```

352         else {
353             this->dispatch_vec_MWh[i] += difference_MWh;
354             dispatchable_MWh += difference_MWh;
355             stored_energy_MWh -= difference_MWh;
356         }
357     }
358 }
359
360 else {
361     this->dispatch_vec_MWh[i] = demand_MWh;
362     dispatchable_MWh += this->dispatch_vec_MWh[i];
363
364     difference_MWh = production_MWh - demand_MWh;
365
366     if (
367         (storage_capacity_MWh > 0) and
368         (stored_energy_MWh < storage_capacity_MWh)
369     ) {
370         room_MWh = storage_capacity_MWh - stored_energy_MWh;
371
372         if (difference_MWh > room_MWh) {
373             stored_energy_MWh += room_MWh;
374         }
375
376         else {
377             stored_energy_MWh += difference_MWh;
378         }
379     }
380 }
381 }
382
383 this->dispatchable_MWh = round(dispatchable_MWh);
384
385 if (this->dispatch_MWh > this->dispatchable_MWh) {
386     this->dispatch_MWh = this->dispatchable_MWh;
387 }
388
389 return;
390 } /* __computeDispatch() */

```

4.12.3.4 __computeProduction()

```

void TidalTurbine::__computeProduction (
    void ) [private]

```

Helper method to compute production values.

```

296 {
297     double production_MWh = 0;
298
299     for (int i = 0; i < 30; i++) {
300         this->production_vec_MWh[i] =
301             this->max_daily_production_MWh * this->capacity_factor_vec[i];
302
303         production_MWh += this->production_vec_MWh[i];
304     }
305
306     this->production_MWh = round(production_MWh);
307
308     return;
309 } /* __computeProduction() */

```

4.12.3.5 __computeProductionCosts()

```

void TidalTurbine::__computeProductionCosts (
    void ) [private]

```

Helper method to compute production costs (O&M) based on current production level.

```

229 {
230     double operation_maintenance_cost =
231         (this->production_MWh * TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
232     this->operation_maintenance_cost = round(operation_maintenance_cost);
233
234     return;
235 } /* __computeProductionCosts() */

```

4.12.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
114 {
115     // 1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
127         this->tile_improvement_sprite_animated[i].setRotation(0);
128
129         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
130
131         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
132         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
133         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
134         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
135     }
136
137     // 2. draw production text
138     std::string production_string = "[W]:  INCREASE DISPATCH\n";
139     production_string             += "[S]:  DECREASE DISPATCH\n";
140     production_string             += "      \n";
141
142     production_string             += "DISPATCH:  ";
143     production_string             += std::to_string(this->dispatch_MWh);
144     production_string             += " MWh (MAX ";
145     production_string             += std::to_string(this->dispatchable_MWh);
146     production_string             += ")\n";
147
148     production_string             += "O&M COST:  ";
149     production_string             += std::to_string(this->operation_maintenance_cost);
150     production_string             += " K\n";
151
152     sf::Text production_text(
153         production_string,
154         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
155         16
156     );
157
158     production_text.setOrigin(production_text.getLocalBounds().width / 2, 0);
159     production_text.setFillColor(MONOCROME_TEXT_GREEN);
160
161     production_text.setPosition(400 + 30, 400 - 45);
162
163     this->render_window_ptr->draw(production_text);
164
165     return;
166 } /* __drawProductionMenu() */
```

4.12.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
531 {
532     // 1. draw power capacity upgrade sprite
533     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
534         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
535         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
536
537         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
538         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
539     }
```

```

540         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
541         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
542
543         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
544         this->tile_improvement_sprite_animated[i].setRotation(0);
545
546         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
547
548         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
549         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
550         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
551         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
552     }
553
554     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
555
556
557     // 2. draw power capacity upgrade text
558     //      16 char line = "
559     std::string power_upgrade_string = "POWER CAPACITY \n";
560     power_upgrade_string += "
561
562     power_upgrade_string += "CAPACITY: ";
563     power_upgrade_string += std::to_string(this->capacity_kW);
564     power_upgrade_string += " kW\n";
565
566     power_upgrade_string += "LEVEL: ";
567     power_upgrade_string += std::to_string(this->upgrade_level);
568     power_upgrade_string += "\n\n";
569
570     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
571         power_upgrade_string += "[W]: + 100 kW (";
572         power_upgrade_string += std::to_string(TIDAL_TURBINE_BUILD_COST);
573         power_upgrade_string += " K)\n";
574     }
575
576     else {
577         power_upgrade_string += " * MAX LEVEL * \n";
578     }
579
580     sf::Text power_upgrade_text = sf::Text(
581         power_upgrade_string,
582         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
583         16
584     );
585
586     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
587     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
588     power_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
589
590     this->render_window_ptr->draw(power_upgrade_text);
591
592
593     // 3. draw energy capacity (storage) upgrade sprite
594     this->render_window_ptr->draw(this->storage_upgrade_sprite);
595     this->render_window_ptr->draw(this->upgrade_plus_sprite);
596
597
598     // 4. draw energy capacity (storage) upgrade text
599     //      16 char line = "
600     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
601     energy_upgrade_string += "
602
603     energy_upgrade_string += "CAPACITY: ";
604     energy_upgrade_string += std::to_string(this->storage_level * 200);
605     energy_upgrade_string += " kWh\n";
606
607     energy_upgrade_string += "LEVEL: ";
608     energy_upgrade_string += std::to_string(this->storage_level);
609     energy_upgrade_string += "\n\n";
610
611     if (this->storage_level < MAX_STORAGE_LEVELS) {
612         energy_upgrade_string += "[D]: + 200 kWh (";
613         energy_upgrade_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
614         energy_upgrade_string += " K)\n";
615     }
616
617     else {
618         energy_upgrade_string += " * MAX LEVEL * \n";
619     }
620
621     sf::Text energy_upgrade_text = sf::Text(
622         energy_upgrade_string,
623         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
624         16
625     );
626

```

```

627     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
628     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
629     energy_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
630
631     this->render_window_ptr->draw(energy_upgrade_text);
632
633     return;
634 } /* __drawUpgradeOptions() */

```

4.12.3.8 __handleKeyPressEvents()

```

void TidalTurbine::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

405 {
406     if (this->just_built) {
407         return;
408     }
409
410     switch (this->event_ptr->key.code) {
411         case (sf::Keyboard::U): {
412             this->__openUpgradeMenu();
413
414             break;
415         }
416
417         case (sf::Keyboard::W): {
418             if (this->production_menu_open) {
419                 this->dispatch_MWh++;
420
421                 if (this->dispatch_MWh > this->dispatchable_MWh) {
422                     this->dispatch_MWh = 0;
423                 }
424
425                 this->__computeProductionCosts();
426                 this->assets_manager_ptr->getSound("interface click")->play();
427             }
428
429             else if (this->upgrade_menu_open) {
430                 this->__upgradePowerCapacity();
431             }
432
433             break;
434         }
435
436         case (sf::Keyboard::S): {
437             if (this->production_menu_open) {
438                 this->dispatch_MWh--;
439
440                 if (this->dispatch_MWh < 0) {
441                     this->dispatch_MWh = this->dispatchable_MWh;
442                 }
443
444                 this->__computeProductionCosts();
445                 this->assets_manager_ptr->getSound("interface click")->play();
446             }
447
448             break;
449         }
450
451         case (sf::Keyboard::D): {
452             if (this->upgrade_menu_open) {
453                 this->__upgradeStorageCapacity();
454                 this->__computeProduction();
455                 this->__computeDispatch();
456             }
457
458             break;
459         }
460
461         default: {
462             // do nothing!
463         }
464     }
465 }

```

```

468         break;
469     }
470 }
471
472 return;
473 } /* __handleKeyPressEvents() */

```

4.12.3.9 __handleMouseButtonEvents()

```

void TidalTurbine::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

488 {
489     if (this->just_built) {
490         return;
491     }
492
493     switch (this->event_ptr->mouseButton.button) {
494         case (sf::Mouse::Left): {
495             //...
496
497             break;
498         }
499
500         case (sf::Mouse::Right): {
501             //...
502
503             break;
504         }
505
506         default: {
507             // do nothing!
508
509             break;
510         }
511     }
512 }
513
514 return;
515 } /* __handleMouseButtonEvents() */

```

4.12.3.10 __sendImprovementStateMessage()

```

void TidalTurbine::__sendImprovementStateMessage (
    void ) [private]

```

Helper method to format and sent improvement state message.

```

649 {
650     Message improvement_state_message;
651
652     improvement_state_message.channel = GAME_CHANNEL;
653     improvement_state_message.subject = "improvement state";
654
655     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
656     improvement_state_message.int_payload["operation_maintenance_cost"] =
657         this->operation_maintenance_cost;
658
659     this->message_hub_ptr->sendMessage(improvement_state_message);
660
661     std::cout << "Improvement state message sent by " << this << std::endl;
662
663     return;
664 } /* __sendImprovementStateMessage() */

```

4.12.3.11 __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() */
```

4.12.3.12 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

```
181 {
182     if (this->credits < TIDAL_TURBINE_BUILD_COST) {
183         std::cout << "Cannot upgrade tidal turbine: insufficient credits (need "
184             << TIDAL_TURBINE_BUILD_COST << " K)" << std::endl;
185
186         this->__sendInsufficientCreditsMessage();
187         return;
188     }
189
190     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
191         return;
192     }
193
194     this->health = 100;
195
196     this->capacity_kW += 100;
197     this->upgrade_level++;
198
199     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
200
201     this->__computeProduction();
202     this->__computeDispatch();
203
204     this->just_upgraded = true;
205
206     this->assets_manager_ptr->getSound("upgrade")->play();
207
208     this->__sendCreditsSpentMessage(TIDAL_TURBINE_BUILD_COST);
209     this->__sendTileStateRequest();
210     this->__sendGameStateRequest();
211
212     return;
213 } /* __upgradePowerCapacity() */
```


4.12.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from [TileImprovement](#).

```
871 {
872     // 1. update
873     this->__computeCapacityFactors();
874     this->update();
875
876     // 2. send improvement state message
877     this->__sendImprovementStateMessage();
878
879     // 3. handle start/stop
880     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
881         this->is_running = true;
882     }
883
884     else if (this->is_running and (this->dispatch_MWh <= 0)) {
885         this->is_running = false;
886     }
887
888     // 4. handle equipment health
889     if (this->is_running) {
890         this->health--;
891
892         if (this->health <= 0) {
893             this->__breakdown();
894         }
895     }
896
897     // 5. send tile state request (if selected)
898     if (this->is_selected) {
899         this->__sendTileStateRequest();
900     }
901
902     return;
903 } /* advanceTurn() */
```

4.12.3.14 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](#).

```
992 {
993     // 1. if just built, call base method and return
994     if (this->just_built) {
995         TileImprovement :: draw();
996
997         return;
998     }
999
1000
1001     // 2. handle upgrade effects
1002     if (this->just_upgraded) {
1003         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1004             this->tile_improvement_sprite_animated[i].setColor(
1005                 sf::Color(
1006                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1007                     255,
1008                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1009                     255
1010                 )
1011             );
1012
1013             this->tile_improvement_sprite_animated[i].setScale(
1014                 sf::Vector2f(
```

```

1015         1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1016         1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1017     )
1018 };
1019 }
1020
1021 this->upgrade_frame++;
1022 }
1023
1024 if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1025     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1026         this->tile_improvement_sprite_animated[i].setColor(
1027             sf::Color(255,255,255,255)
1028         );
1029
1030         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1031     }
1032
1033     this->just_upgraded = false;
1034     this->upgrade_frame = 0;
1035 }
1036
1037 // 3. handle bobbing
1038 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1039     this->tile_improvement_sprite_animated[i].setPosition(
1040         this->position_x,
1041         this->position_y + this->bobbing_y * cos(
1042             (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1043         )
1044     );
1045 }
1046
1047 // 4. draw first element of animated sprite
1048 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1049
1050 // 5. draw second element of animated sprite
1051 if (this->is_running) {
1052     this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1053 }
1054
1055 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1056
1057 // 6. draw storage upgrades
1058 for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1059     this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1060 }
1061
1062 // 7. handle dispatch illustration
1063 if (this->dispatch_MWh > 0) {
1064     this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1065     this->__drawDispatch();
1066 }
1067
1068 // 8. draw production menu
1069 if (this->production_menu_open) {
1070     this->render_window_ptr->draw(this->production_menu_backing);
1071     this->render_window_ptr->draw(this->production_menu_backing_text);
1072
1073     this->__drawProductionMenu();
1074 }
1075
1076 // 9. draw upgrade menu
1077 if (this->upgrade_menu_open) {
1078     this->render_window_ptr->draw(this->upgrade_menu_backing);
1079     this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1080
1081     this->__drawUpgradeOptions();
1082 }
1083
1084 // 10. handle broken effects
1085 if (this->is_broken) {
1086     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1087         this->tile_improvement_sprite_animated[i].setColor(
1088             sf::Color(
1089                 255,
1090                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1091                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1092                 255
1093             )
1094         );
1095     }
1096 }

```

```

1102         );
1103     }
1104 }
1105
1106     this->frame++;
1107     return;
1108 } /* draw() */

```

4.12.3.15 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

782 {
783     //          32 char x 17 line console "-----\n";
784     std::string options_substring = "CAPACITY: ";
785     options_substring += std::to_string(this->capacity_kW);
786     options_substring += " kW (level ";
787     options_substring += std::to_string(this->upgrade_level);
788     options_substring += ")\n";
789
790     options_substring += "PRODUCTION: ";
791     options_substring += std::to_string(this->production_MWh);
792     options_substring += " MWh\n";
793
794     options_substring += "DISPATCHABLE: ";
795     options_substring += std::to_string(this->dispatchable_MWh);
796     options_substring += " MWh\n";
797
798     options_substring += "HEALTH: ";
799     options_substring += std::to_string(this->health);
800     options_substring += "/100";
801
802     if (this->health <= 0) {
803         options_substring += " ** BROKEN! **\n";
804     }
805
806     else {
807         options_substring += "\n";
808     }
809
810     options_substring += "
811     options_substring += "**** TIDAL TURBINE OPTIONS ****
812     options_substring += "
813
814     options_substring += "      [E]: ";
815
816     if (this->is_broken) {
817         options_substring += "*** BROKEN! ***\n";
818     }
819
820     else {
821         options_substring += "OPEN PRODUCTION MENU\n";
822     }
823
824     options_substring += "      [U]: OPEN UPGRADE MENU
825     options_substring += "HOLD [P]: SCRAP ("
826     options_substring += std::to_string(SCRAP_COST);
827     options_substring += " K)\n";
828
829     return options_substring;
830 } /* getTileOptionsSubstring() */

```

4.12.3.16 processEvent()

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

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

Reimplemented from [TileImprovement](#).

```
943 {
944     TileImprovement :: processEvent();
945
946     if (this->event_ptr->type == sf::Event::KeyPressed) {
947         this->__handleKeyPressEvents();
948     }
949
950     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
951         this->__handleMouseButtonEvents();
952     }
953
954     return;
955 } /* processEvent() */
```

4.12.3.17 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
970 {
971     TileImprovement :: processMessage();
972
973     //...
974
975     return;
976 } /* processMessage() */
```

4.12.3.18 setIsSelected()

```
void TidalTurbine::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](#).

```
847 {
848     TileImprovement :: setIsSelected(is_selected);
849
850     if (this->is_running and this->is_selected) {
851         this->assets_manager_ptr->getSound("water flow")->play();
852     }
853 }
```

```
854     return;  
855 } /* setIsSelected() */
```

4.12.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from [TileImprovement](#).

```
918 {  
919     this->__computeProduction();  
920     this->__computeProductionCosts();  
921     this->__computeDispatch();  
922  
923     if (this->is_selected) {  
924         this->__sendTileStateRequest();  
925     }  
926  
927     return;  
928 } /* update() */
```

4.12.4 Member Data Documentation

4.12.4.1 bobbing_y

```
double TidalTurbine::bobbing_y
```

The bobbing extent of the tidal turbine.

4.12.4.2 capacity_factor_vec

```
std::vector<double> TidalTurbine::capacity_factor_vec
```

A vector of daily capacity factors for the current month.

4.12.4.3 capacity_kW

```
int TidalTurbine::capacity_kW
```

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

4.12.4.4 dispatch_MWh

```
int TidalTurbine::dispatch_MWh
```

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

4.12.4.5 dispatch_vec_MWh

```
std::vector<double> TidalTurbine::dispatch_vec_MWh
```

A vector of daily dispatch [MWh] for the current month.

4.12.4.6 dispatchable_MWh

```
int TidalTurbine::dispatchable_MWh
```

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

4.12.4.7 max_daily_production_MWh

```
double TidalTurbine::max_daily_production_MWh
```

The maximum daily production [MWh] of the solar PV array.

4.12.4.8 production_MWh

```
int TidalTurbine::production_MWh
```

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

4.12.4.9 production_vec_MWh

```
std::vector<double> TidalTurbine::production_vec_MWh
```

A vector of daily production [MWh] for the current month.

4.12.4.10 rotor_drotation

```
double TidalTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

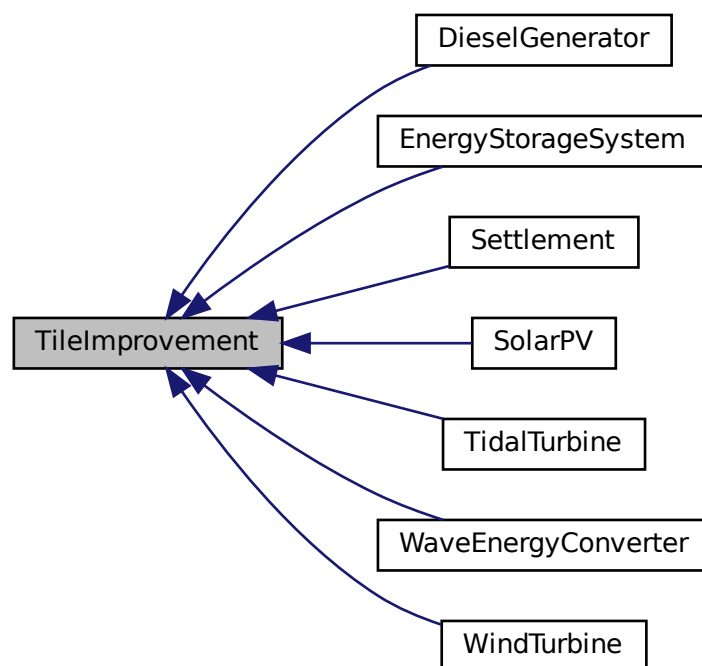
- header/[TidalTurbine.h](#)
- source/[TidalTurbine.cpp](#)

4.13 TileImprovement Class Reference

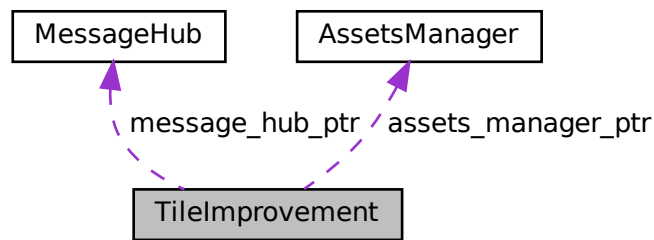
A base class for the tile improvement hierarchy.

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

Inheritance diagram for TileImprovement:



Collaboration diagram for TileImprovement:



Public Member Functions

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

- bool `is_broken`
A boolean which indicated whether or not improvement is broken.
- unsigned long long int `frame`
The current frame of this object.
- int `credits`
The current balance of credits.
- int `month`
The current month of play.
- int `demand_MWh`
The current demand [MWh].
- 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_kWh`
The rated energy capacity [kWh] of the storage.
- int `storage_level`
The level of storage installed alongside the tile improvement.
- int `operation_maintenance_cost`
The operation and maintenance costs for this turn.
- int `tile_resource`
The renewable resource quality of the tile.
- double `tile_resource_scalar`
A scalar associated with the renewable resource quality.
- double `position_x`
The x position of the tile improvement.
- double `position_y`
The y position of the tile improvement.
- std::vector< double > `demand_vec_MWh`
A vector of daily demands [MWh] for the current month.
- 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.
 - sf::CircleShape [dispatch_backing](#)
A backing circle for dispatch text illustration.
 - sf::Text [dispatch_text](#)
Text for illustrating dispatch.

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 [__setUpDispatchIllustration](#) (void)
Helper method to set up and position dispatch 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 [__breakdown](#) (void)
Helper method to trigger an equipment breakdown.
- void [__repair](#) (void)
Helper method to repair a tile improvement.
- 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.
- void [__drawDispatch](#) (void)
Helper method to draw dispatch illustration.

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,
    int tile_resource,
    sf::Event * event_ptr,
    sf::RenderWindow * render_window_ptr,
    AssetsManager * assets_manager_ptr,
    MessageHub * message_hub_ptr )
```

Constructor for the [TileImprovement](#) class.

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

Parameters

<i>position_x</i>	The x position of the tile.
<i>position_y</i>	The y position of the tile.
<i>tile_resource</i>	The renewable resource quality 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.

```
716 {
717     // 1. set attributes
718
719     // 1.1. protected
720     this->event_ptr = event_ptr;
721     this->render_window_ptr = render_window_ptr;
722 }
```

```

723     this->assets_manager_ptr = assets_manager_ptr;
724     this->message_hub_ptr = message_hub_ptr;
725
726     // 1.2. public
727     this->is_selected = true;
728     this->just_built = true;
729     this->production_menu_open = false;
730     this->upgrade_menu_open = false;
731     this->is_broken = false;
732
733     this->just_upgraded = false;
734     this->upgrade_frame = 0;
735
736     this->frame = 0;
737     this->credits = 0;
738     this->month = 1;
739     this->demand_MWh = 0;
740
741     this->demand_vec_MWh.resize(30, 0);
742
743     this->operation_maintenance_cost = 0;
744
745     this->tile_resource = tile_resource;
746
747     switch (this->tile_resource) {
748     case (0): {
749         this->tile_resource_scalar = 0.7;
750
751         break;
752     }
753
754     case (1): {
755         this->tile_resource_scalar = 0.85;
756
757         break;
758     }
759
760     case (2): {
761         this->tile_resource_scalar = 1;
762
763         break;
764     }
765
766     case (3): {
767         this->tile_resource_scalar = 1.15;
768
769         break;
770     }
771
772     case (4): {
773         this->tile_resource_scalar = 1.3;
774
775         break;
776     }
777
778     default: {
779         this->tile_resource_scalar = 1;
780     }
781 }
782
783 this->position_x = position_x;
784 this->position_y = position_y;
785
786 this->game_phase = "build settlement";
787
788 this->__setUpProductionMenu();
789 this->__setUpUpgradeMenu();
790 this->__setUpDispatchIllustration();
791
792 std::cout << "TileImprovement constructed at " << this << std::endl;
793
794 return;
795 }
796
797 /* TileImprovement() */

```

4.13.2.2 ~TileImprovement()

```
TileImprovement::~TileImprovement (
```

```
void ) [virtual]
```

Destructor for the [TileImprovement](#) class.

```
1032 {
1033     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
1034
1035     return;
1036 } /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
421 {
422     this->is_broken = true;
423     this->is_running = false;
424     this->assets_manager_ptr->getSound("breakdown")->play();
425
426     return;
427 } /* __breakdown() */
```

4.13.3.2 __closeProductionMenu()

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

Helper method to close the production menu.

```
397 {
398     if (not this->production_menu_open) {
399         return;
400     }
401
402     this->production_menu_open = false;
403     this->assets_manager_ptr->getSound("build menu close")->play();
404
405     return;
406 } /* __closeProductionMenu() */
```

4.13.3.3 __closeUpgradeMenu()

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

Helper method to close the build menu.

```
506 {
507     if (not this->upgrade_menu_open) {
508         return;
509     }
510
511     this->upgrade_menu_open = false;
512     this->assets_manager_ptr->getSound("build menu close")->play();
513
514     return;
515 } /* __closeUpgradeMenu() */
```

4.13.3.4 __drawDispatch()

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

Helper method to draw dispatch illustration.

```
637 {
638     double alpha = 255 * pow(cos((0.5 * M_PI * this->frame) / FRAMES_PER_SECOND), 2);
639
640
641     // 1. dispatch backing
642     sf::Color backing_colour = this->dispatch_backing.getFillColor();
643
644     backing_colour.a = alpha;
645
646     this->dispatch_backing.setFillColor(backing_colour);
647     this->dispatch_backing.setOutlineColor(sf::Color(0, 0, 0, alpha));
648
649     this->render_window_ptr->draw(this->dispatch_backing);
650
651
652     // 2. dispatch text
653     this->dispatch_text.setOrigin(
654         this->dispatch_text.getLocalBounds().width / 2,
655         this->dispatch_text.getLocalBounds().height / 2
656     );
657
658     this->dispatch_text.setFillColor(
659         sf::Color(0, 0, 0, alpha)
660     );
661
662     this->render_window_ptr->draw(this->dispatch_text);
663
664     return;
665 } /* __drawDispatch() */
```

4.13.3.5 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
277 {
278     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
279         return;
280     }
281
282     if (this->just_built) {
283         return;
284     }
285
286     switch (this->event_ptr->key.code) {
287         case (sf::Keyboard::E): {
288             this->__openProductionMenu();
289
290             break;
291         }
292
293
294         default: {
295             // do nothing!
296
297             break;
298         }
299     }
300
301     return;
302 } /* __handleKeyPressEvents() */
```

4.13.3.6 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
317 {
318     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
319         return;
320     }
321
322     if (this->just_built) {
323         return;
324     }
325
326     switch (this->event_ptr->mouseButton.button) {
327         case (sf::Mouse::Left): {
328             //...
329
330             break;
331         }
332
333         case (sf::Mouse::Right): {
334             //...
335
336             break;
337         }
338     }
339
340     default: {
341         // do nothing!
342
343         break;
344     }
345 }
346
347 return;
348 }
349 } /* __handleMouseButtonEvents() */
```

4.13.3.7 __openProductionMenu()

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

Helper method to open the production menu.

```
364 {
365     if (this->is_broken) {
366         this->assets_manager_ptr->getSound("breakdown")->play();
367         return;
368     }
369
370     if (this->production_menu_open) {
371         return;
372     }
373
374     if (this->upgrade_menu_open) {
375         this->__closeUpgradeMenu();
376     }
377
378     this->production_menu_open = true;
379     this->assets_manager_ptr->getSound("build menu open")->play();
380
381     return;
382 } /* __openProductionMenu() */
```

4.13.3.8 __openUpgradeMenu()

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

Helper method to open the upgrade menu.

```
478 {
479     if (this->upgrade_menu_open) {
480         return;
481     }
482     if (this->production_menu_open) {
483         this->__closeProductionMenu();
484     }
485     this->upgrade_menu_open = true;
486     this->assets_manager_ptr->getSound("build menu open")->play();
487     return;
488 }
489
490 /* __openUpgradeMenu() */
```

4.13.3.9 __repair()

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

Helper method to repair a tile improvement.

```
442 {
443     this->health = 100;
444     if (this->is_broken) {
445         this->is_broken = false;
446         this->assets_manager_ptr->getSound("positive notification")->play();
447     }
448     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
449         this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
450     }
451     else {
452         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
453             this->tile_improvement_sprite_animated[i].setColor(
454                 sf::Color(255, 255, 255, 255)
455             );
456         }
457     }
458     return;
459 }
460
461 /* __repair() */
```

4.13.3.10 __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.
----------------------	--


```

583 {
584     Message credits_spent_message;
585
586     credits_spent_message.channel = GAME_CHANNEL;
587     credits_spent_message.subject = "credits spent";
588
589     credits_spent_message.int_payload["credits spent"] = credits_spent;
590
591     this->message_hub_ptr->sendMessage(credits_spent_message);
592
593     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
594               << std::endl;
595     return;
596 } /* __sendCreditsSpentMessage() */

```

4.13.3.11 __sendGameStateRequest()

```

void TileImprovement::__sendGameStateRequest (
    void ) [protected]

```

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

```

556 {
557     Message game_state_request;
558
559     game_state_request.channel = GAME_CHANNEL;
560     game_state_request.subject = "state request";
561
562     this->message_hub_ptr->sendMessage(game_state_request);
563
564     std::cout << "Game state request message sent by " << this << std::endl;
565     return;
566 } /* __sendGameStateRequest() */

```

4.13.3.12 __sendInsufficientCreditsMessage()

```

void TileImprovement::__sendInsufficientCreditsMessage (
    void ) [protected]

```

Helper method to format and send an insufficient credits message.

```

611 {
612     Message insufficient_credits_message;
613
614     insufficient_credits_message.channel = GAME_CHANNEL;
615     insufficient_credits_message.subject = "insufficient credits";
616
617     this->message_hub_ptr->sendMessage(insufficient_credits_message);
618
619     std::cout << "Insufficient credits message sent by " << this << std::endl;
620
621     return;
622 } /* __sendInsufficientCreditsMessage() */

```

4.13.3.13 __sendTileStateRequest()

```

void TileImprovement::__sendTileStateRequest (
    void ) [protected]

```

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

```

531 {
532     Message tile_state_request;
533
534     tile_state_request.channel = TILE_STATE_CHANNEL;
535     tile_state_request.subject = "state request";
536
537     this->message_hub_ptr->sendMessage(tile_state_request);
538
539     std::cout << "Tile state request sent by " << this << std::endl;
540     return;
541 } /* __sendTileStateRequest() */

```

4.13.3.14 __setUpDispatchIllustration()

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

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

```
178 {
179     // 1. set up backing
180     this->dispatch_backing.setRadius(16);
181
182     this->dispatch_backing.setOrigin(
183         this->dispatch_backing.getLocalBounds().width / 2,
184         this->dispatch_backing.getLocalBounds().height / 2
185     );
186
187     this->dispatch_backing.setPosition(
188         this->position_x,
189         this->position_y
190     );
191
192     this->dispatch_backing.setFillColor(RESOURCE_CHIP_GREY);
193     this->dispatch_backing.setOutlineThickness(1);
194     this->dispatch_backing.setOutlineColor(sf::Color(0, 0, 0, 255));
195
196
197     // 2. set up text
198     this->dispatch_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
199     this->dispatch_text.setFillColor(sf::Color(0, 0, 0, 255));
200     this->dispatch_text.setCharacterSize(16);
201     this->dispatch_text.setPosition(
202         this->position_x,
203         this->position_y - 4
204     );
205
206     return;
207 } /* __setUpDispatchIllustration() */
```

4.13.3.15 __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(MONOCROME_SCREEN_BACKGROUND);
74     this->production_menu_backing.setOutlineColor(MENU_FRAME_GREY);
75     this->production_menu_backing.setOutlineThickness(4);
76
77     this->production_menu_backing_text.setString("**** PRODUCTION MENU ****");
78     this->production_menu_backing_text.setFont(
79         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
80     );
81     this->production_menu_backing_text.setCharacterSize(16);
82     this->production_menu_backing_text.setFillColor(MONOCROME_TEXT_GREEN);
83     this->production_menu_backing_text.setOrigin(
84         this->production_menu_backing_text.getLocalBounds().width / 2, 0
85     );
86     this->production_menu_backing_text.setPosition(400, 400 - 128 + 4);
87
88     return;
89 } /* __setUpProductionMenu() */
```

4.13.3.16 __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(MONOCHROME_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() */
```

4.13.3.17 __upgradeStorageCapacity()

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

Helper method to upgrade storage capacity.

```
222 {
223     if (this->credits < ENERGY_STORAGE_SYSTEM_BUILD_COST) {
```

```

224         std::cout << "Cannot add energy storage: insufficient credits (need "
225             << ENERGY_STORAGE_SYSTEM_BUILD_COST << " K)" << std::endl;
226
227         this->__sendInsufficientCreditsMessage();
228         return;
229     }
230
231     if (this->storage_level >= MAX_STORAGE_LEVELS) {
232         return;
233     }
234
235     this->storage_level++;
236     this->storage_kWh += 200;
237
238     this->storage_upgrade_sprite_vec.push_back(
239         sf::Sprite(
240             *(this->assets_manager_ptr->getTexture("storage_level"))
241         )
242     );
243
244     this->storage_upgrade_sprite_vec.back().setOrigin(
245         this->storage_upgrade_sprite_vec.back().getLocalBounds().width / 2,
246         this->storage_upgrade_sprite_vec.back().getLocalBounds().height
247     );
248
249     this->storage_upgrade_sprite_vec.back().setPosition(
250         this->position_x + 18,
251         this->position_y + 25 - 7 * this->storage_upgrade_sprite_vec.size()
252     );
253
254     this->just_upgraded = true;
255
256     this->assets_manager_ptr->getSound("upgrade")->play();
257
258     this->__sendCreditsSpentMessage(ENERGY_STORAGE_SYSTEM_BUILD_COST);
259     this->__sendTileStateRequest();
260
261     return;
262 } /* __upgradeStorageCapacity() */

```

4.13.3.18 advanceTurn()

```

virtual void TileImprovement::advanceTurn (
    void ) [inline], [virtual]

```

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

```
191 {return;}
```

4.13.3.19 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](#).

```

903 {
904     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
905         int alpha = this->tile_improvement_sprite_static.getColor().a;
906
907         alpha += 0.08 * FRAMES_PER_SECOND;
908
909         this->tile_improvement_sprite_static.setColor(
910             sf::Color(255, 255, 255, alpha)
911         );
912
913         this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);

```

```

914
915     if (
916         (alpha >= 255) or
917         (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
918     ) {
919         this->tile_improvement_sprite_static.setColor(
920             sf::Color(255, 255, 255, 255)
921         );
922
923         this->tile_improvement_sprite_static.setPosition(
924             this->position_x,
925             this->position_y + 12
926         );
927
928         this->just_built = false;
929         this->assets_manager_ptr->getSound("place improvement")->play();
930     }
931
932     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
933 }
934
935 else {
936     int alpha = 0;
937
938     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
939         alpha = this->tile_improvement_sprite_animated[i].getColor().a;
940
941         alpha += 0.08 * FRAMES_PER_SECOND;
942
943         this->tile_improvement_sprite_animated[i].setColor(
944             sf::Color(255, 255, 255, alpha)
945         );
946
947         this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
948
949         if (
950             (alpha >= 255) or
951             (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
952         ) {
953             this->tile_improvement_sprite_animated[i].setColor(
954                 sf::Color(255, 255, 255, 255)
955             );
956
957             this->tile_improvement_sprite_animated[i].setPosition(
958                 this->position_x,
959                 this->position_y + 12
960             );
961         }
962
963         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
964     }
965
966     if (
967         (alpha >= 255) or
968         (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
969     ) {
970         this->just_built = false;
971         this->assets_manager_ptr->getSound("place improvement")->play();
972
973         switch (this->tile_improvement_type) {
974             case (TileImprovementType :: WIND_TURBINE): {
975                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
976                     this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
977                     this->tile_improvement_sprite_animated[i].move(0, -32);
978                 }
979
980                 break;
981             }
982
983             case (TileImprovementType :: TIDAL_TURBINE): {
984                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
985                     this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
986                     this->tile_improvement_sprite_animated[i].move(0, -19);
987                 }
988
989                 break;
990             }
991
992             case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
993                 for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
994                     this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
995                     this->tile_improvement_sprite_animated[i].move(0, -32);
996                 }
997             }
998         }
999
1000

```

```

1001             break;
1002         }
1003
1004
1005         default: {
1006             // do nothing!
1007
1008             break;
1009         }
1010     }
1011 }
1012 }
1013
1014
1015     this->frame++;
1016     return;
1017 } /* draw() */

```

4.13.3.20 `getTileOptionsSubstring()`

```

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

```

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

```

195 {return "";}

```

4.13.3.21 `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](#).

```

844 {
845     if (this->event_ptr->type == sf::Event::KeyPressed) {
846         this->__handleKeyPressEvents();
847     }
848
849     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
850         this->__handleMouseButtonEvents();
851     }
852
853     return;
854 } /* processEvent() */

```

4.13.3.22 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](#).

```
869 {
870     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
871         Message game_state_message = this->message_hub_ptr->receiveMessage(
872             GAME_STATE_CHANNEL
873         );
874
875         if (game_state_message.subject == "turn advance") {
876             this->credits = game_state_message.int_payload["credits"];
877             this->month = game_state_message.int_payload["month"];
878             this->demand_MWh = game_state_message.int_payload["demand_MWh"];
879
880             this->advanceTurn();
881
882             std::cout << "Turn advance message read and passed by " << this << std::endl;
883         }
884     }
885
886     return;
887 } /* processMessage() */
```

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

```
817 {
818     this->is_selected = is_selected;
819
820     if ((not is_selected) and this->production_menu_open) {
821         this->__closeProductionMenu();
822     }
823
824     if ((not is_selected) and this->upgrade_menu_open) {
825         this->__closeUpgradeMenu();
826     }
827
828     return;
829 } /* setIsSelected() */
```

4.13.3.24 update()

```
virtual void TileImprovement::update (
    void ) [inline], [virtual]
```

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

```
193 {return;}
```

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 `demand_MWh`

`int` `TileImprovement::demand_MWh`

The current demand [MWh].

4.13.4.4 `demand_vec_MWh`

`std::vector<double>` `TileImprovement::demand_vec_MWh`

A vector of daily demands [MWh] for the current month.

4.13.4.5 `dispatch_backing`

`sf::CircleShape` `TileImprovement::dispatch_backing`

A backing circle for dispatch text illustration.

4.13.4.6 dispatch_text

```
sf::Text TileImprovement::dispatch_text
```

Text for illustrating dispatch.

4.13.4.7 event_ptr

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

A pointer to the event class.

4.13.4.8 frame

```
unsigned long long int TileImprovement::frame
```

The current frame of this object.

4.13.4.9 game_phase

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

The current phase of the game.

4.13.4.10 health

```
int TileImprovement::health
```

The health of the improvement.

4.13.4.11 is_broken

```
bool TileImprovement::is_broken
```

A boolean which indicated whether or not improvement is broken.

4.13.4.12 is_running

```
bool TileImprovement::is_running
```

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

4.13.4.13 is_selected

```
bool TileImprovement::is_selected
```

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

4.13.4.14 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.15 just_upgraded

```
bool TileImprovement::just_upgraded
```

A boolean which indicates that the improvement was just upgraded.

4.13.4.16 message_hub_ptr

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

A pointer to the message hub.

4.13.4.17 month

```
int TileImprovement::month
```

The current month of play.

4.13.4.18 operation_maintenance_cost

```
int TileImprovement::operation_maintenance_cost
```

The operation and maintenance costs for this turn.

4.13.4.19 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.20 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.21 production_menu_backing

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

A backing for the production menu.

4.13.4.22 production_menu_backing_text

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

Text for the production menu backing.

4.13.4.23 production_menu_open

```
bool TileImprovement::production_menu_open
```

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

4.13.4.24 render_window_ptr

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

A pointer to the render window.

4.13.4.25 storage_kWh

```
int TileImprovement::storage_kWh
```

The rated energy capacity [kWh] of the storage.

4.13.4.26 storage_level

```
int TileImprovement::storage_level
```

The level of storage installed alongside the tile improvement.

4.13.4.27 storage_upgrade_sprite

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

A sprite for illustrating storage (in upgrade menu).

4.13.4.28 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).

4.13.4.29 tile_improvement_sprite_animated

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

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

4.13.4.30 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.31 tile_improvement_string

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

A string representation of the tile improvement type.

4.13.4.32 tile_improvement_type

```
TileImprovementType TileImprovement::tile_improvement_type
```

The type of the tile improvement.

4.13.4.33 tile_resource

```
int TileImprovement::tile_resource
```

The renewable resource quality of the tile.

4.13.4.34 tile_resource_scalar

```
double TileImprovement::tile_resource_scalar
```

A scalar associated with the renewable resource quality.

4.13.4.35 upgrade_arrow_sprite

```
sf::Sprite TileImprovement::upgrade_arrow_sprite
```

An upgrade arrow sprite.

4.13.4.36 upgrade_frame

```
int TileImprovement::upgrade_frame
```

The frame of the upgrade animation.

4.13.4.37 upgrade_level

```
int TileImprovement::upgrade_level
```

The upgrade level of the improvement.

4.13.4.38 upgrade_menu_backing

```
sf::RectangleShape TileImprovement::upgrade_menu_backing
```

A backing for the upgrade menu.

4.13.4.39 upgrade_menu_backing_text

```
sf::Text TileImprovement::upgrade_menu_backing_text
```

Text for the upgrade menu backing.

4.13.4.40 upgrade_menu_open

```
bool TileImprovement::upgrade_menu_open
```

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

4.13.4.41 upgrade_plus_sprite

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

An upgrade plus sprite.

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

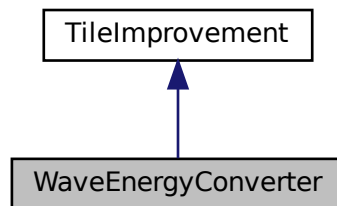
- header/[TileImprovement.h](#)
- source/[TileImprovement.cpp](#)

4.14 WaveEnergyConverter Class Reference

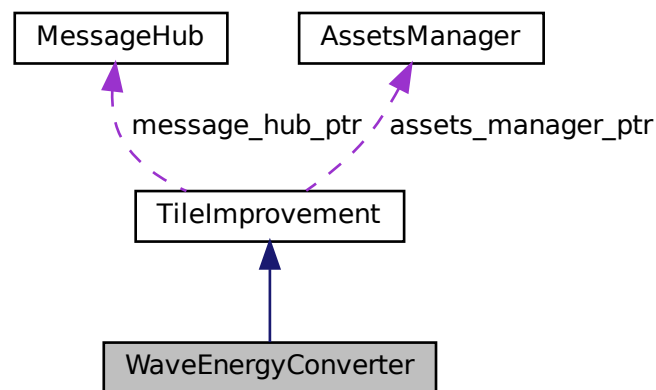
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for WaveEnergyConverter:



Collaboration diagram for WaveEnergyConverter:



Public Member Functions

- [WaveEnergyConverter](#) (double, double, int, 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 [setIsSelected](#) (bool)
Method to set the is selected attribute.

- void [advanceTurn](#) (void)
Method to handle turn advance.
- void [update](#) (void)
Method to trigger production and dispatchable updates.
- 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 [dispatch_MWh](#)
The current dispatch [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).
- double [max_daily_production_MWh](#)
The maximum daily production [MWh] of the solar PV array.
- double [bobbing_y](#)
The bobbing extent of the wave energy converter.
- std::vector< double > [capacity_factor_vec](#)
A vector of daily capacity factors for the current month.
- std::vector< double > [production_vec_MWh](#)
A vector of daily production [MWh] for the current month.
- std::vector< double > [dispatch_vec_MWh](#)
A vector of daily dispatch [MWh] for the current month.

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
Helper method to set up tile improvement sprite (static).
- void [__drawProductionMenu](#) (void)
Helper method to draw production menu assets.
- void [__upgradePowerCapacity](#) (void)
Helper method to upgrade power capacity.
- void [__computeProductionCosts](#) (void)
Helper method to compute production costs (O&M) based on current production level.
- void [__breakdown](#) (void)
Helper method to trigger an equipment breakdown.
- void [__computeCapacityFactors](#) (void)
Helper method to compute capacity factors.
- void [__computeProduction](#) (void)

- Helper method to compute production values.*
- void [__computeDispatch](#) (void)
- Helper method to compute dispatch values.*
- 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.*
- void [__sendImprovementStateMessage](#) (void)
- Helper method to format and sent improvement state message.*

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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.

```

730 :
731 TileImprovement (
732     position_x,
733     position_y,
734     tile_resource,
735     event_ptr,
736     render_window_ptr,
737     assets_manager_ptr,
738     message_hub_ptr
739 )
740 {
741     // 1. set attributes
742
743     // 1.1. private
744     //...
745
746     // 1.2. public
747     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
748
749     this->is_running = false;
750
751     this->health = 100;
752
753     this->capacity_kW = 100;
754     this->upgrade_level = 1;
755
756     this->storage_kWh = 0;
757     this->storage_level = 0;
758
759     this->production_MWh = 0;
760     this->dispatch_MWh = 0;
761     this->dispatchable_MWh = 0;
762
763     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
764
765     this->bobbing_y = 4;
766
767     this->capacity_factor_vec.resize(30, 0);
768     this->production_vec_MWh.resize(30, 0);
769     this->dispatch_vec_MWh.resize(30, 0);
770
771     this->tile_improvement_string = "WAVE ENERGY";
772
773     this->__setUpTileImprovementSpriteAnimated();
774     this->__computeCapacityFactors();
775     this->update();
776
777     std::cout << "WaveEnergyConverter constructed at " << this << std::endl;
778
779     return;
780 } /* WaveEnergyConverter() */

```

4.14.2.2 ~WaveEnergyConverter()

```

WaveEnergyConverter::~WaveEnergyConverter (
    void ) [virtual]

```

Destructor for the [WaveEnergyConverter](#) class.

```

1149 {
1150     std::cout << "WaveEnergyConverter at " << this << " destroyed" << std::endl;
1151
1152     return;
1153 } /* ~WaveEnergyConverter() */

```

4.14.3 Member Function Documentation

4.14.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
250 {
251     TileImprovement :: __breakdown();
252
253     this->production_MWh = 0;
254     this->dispatch_MWh = 0;
255     this->dispatchable_MWh = 0;
256     this->operation_maintenance_cost = 0;
257
258     return;
259 } /* __breakdown() */
```

4.14.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
274 {
275     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
276     std::default_random_engine generator(seed);
277
278     double mean =
279         this->tile_resource_scalar * MEAN_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
280
281     double stdev = STDEV_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
282
283     if (this->tile_resource_scalar > 1) {
284         stdev /= this->tile_resource_scalar;
285     }
286
287     std::normal_distribution<double> normal_dist(mean, stdev);
288
289     double capacity_factor = 0;
290
291     for (int i = 0; i < 30; i++) {
292         capacity_factor = normal_dist(generator);
293
294         if (capacity_factor < 0) {
295             capacity_factor = 0;
296         }
297
298         this->capacity_factor_vec[i] = capacity_factor;
299     }
300
301     return;
302 } /* __computeCapacityFactors() */
```

4.14.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
345 {
346     double stored_energy_MWh = 0;
347     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
348
349     double demand_MWh = 0;
350     double production_MWh = 0;
351     double dispatchable_MWh = 0;
```

```

352     double difference_MWh = 0;
353
354     double room_MWh = 0;
355
356     for (int i = 0; i < 30; i++) {
357         demand_MWh = this->demand_vec_MWh[i];
358         production_MWh = this->production_vec_MWh[i];
359
360         if (production_MWh <= demand_MWh) {
361             this->dispatch_vec_MWh[i] = production_MWh;
362             dispatchable_MWh += this->dispatch_vec_MWh[i];
363
364             difference_MWh = demand_MWh - production_MWh;
365
366             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
367                 if (difference_MWh > stored_energy_MWh) {
368                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
369                     dispatchable_MWh += stored_energy_MWh;
370                     stored_energy_MWh = 0;
371                 }
372
373                 else {
374                     this->dispatch_vec_MWh[i] += difference_MWh;
375                     dispatchable_MWh += difference_MWh;
376                     stored_energy_MWh -= difference_MWh;
377                 }
378             }
379         }
380
381         else {
382             this->dispatch_vec_MWh[i] = demand_MWh;
383             dispatchable_MWh += this->dispatch_vec_MWh[i];
384
385             difference_MWh = production_MWh - demand_MWh;
386
387             if (
388                 (storage_capacity_MWh > 0) and
389                 (stored_energy_MWh < storage_capacity_MWh)
390             ) {
391                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
392
393                 if (difference_MWh > room_MWh) {
394                     stored_energy_MWh += room_MWh;
395                 }
396
397                 else {
398                     stored_energy_MWh += difference_MWh;
399                 }
400             }
401         }
402     }
403
404     this->dispatchable_MWh = round(dispatchable_MWh);
405
406     if (this->dispatch_MWh > this->dispatchable_MWh) {
407         this->dispatch_MWh = this->dispatchable_MWh;
408     }
409
410     return;
411 } /* __computeDispatch() */

```

4.14.3.4 __computeProduction()

```

void WaveEnergyConverter::__computeProduction (
    void ) [private]

```

Helper method to compute production values.

```

317 {
318     double production_MWh = 0;
319
320     for (int i = 0; i < 30; i++) {
321         this->production_vec_MWh[i] =
322             this->max_daily_production_MWh * this->capacity_factor_vec[i];
323
324         production_MWh += this->production_vec_MWh[i];
325     }
326
327     this->production_MWh = round(production_MWh);

```

```

328
329     return;
330 } /* __computeProduction() */

```

4.14.3.5 __computeProductionCosts()

```

void WaveEnergyConverter::__computeProductionCosts (
    void ) [private]

```

Helper method to compute production costs (O&M) based on current production level.

```

229 {
230     double operation_maintenance_cost =
231         (this->production_MWh * WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
232     this->operation_maintenance_cost = round(operation_maintenance_cost);
233
234     return;
235 } /* __computeProductionCosts() */

```

4.14.3.6 __drawProductionMenu()

```

void WaveEnergyConverter::__drawProductionMenu (
    void ) [private]

```

Helper method to draw production menu assets.

```

114 {
115     // 1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
127         this->tile_improvement_sprite_animated[i].setRotation(0);
128
129         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
130
131         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
132         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
133         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
134         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
135     }
136
137     // 2. draw production text
138     std::string production_string = "[W]: INCREASE DISPATCH\n";
139     production_string += "[S]: DECREASE DISPATCH\n";
140     production_string += "\n";
141
142     production_string += "DISPATCH: ";
143     production_string += std::to_string(this->dispatch_MWh);
144     production_string += " MWh (MAX ";
145     production_string += std::to_string(this->dispatchable_MWh);
146     production_string += ")\n";
147
148     production_string += "O&M COST: ";
149     production_string += std::to_string(this->operation_maintenance_cost);
150     production_string += " K\n";
151
152     sf::Text production_text(
153         production_string,
154         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
155         16
156     );
157
158     production_text.setOrigin(production_text.getLocalBounds().width / 2, 0);

```

```

159     production_text.setFillColor(MONOCROME_TEXT_GREEN);
160
161     production_text.setPosition(400 + 30, 400 - 45);
162
163     this->render_window_ptr->draw(production_text);
164
165     return;
166 } /* __drawProductionMenu() */

```

4.14.3.7 __drawUpgradeOptions()

```

void WaveEnergyConverter::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

551 {
552     // 1. draw power capacity upgrade sprite
553     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
554         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
555         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
556
557         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
558         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
559
560         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
561         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
562
563         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
564
565         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
566         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
567         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
568     }
569
570     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
571
572
573     // 2. draw power capacity upgrade text
574     // 16 char line = "\n"
575     std::string power_upgrade_string = "POWER CAPACITY \n";
576     power_upgrade_string += "\n";
577
578     power_upgrade_string += "CAPACITY: ";
579     power_upgrade_string += std::to_string(this->capacity_kW);
580     power_upgrade_string += " kW\n";
581
582     power_upgrade_string += "LEVEL: ";
583     power_upgrade_string += std::to_string(this->upgrade_level);
584     power_upgrade_string += "\n\n";
585
586     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
587         power_upgrade_string += "[W]: + 100 kW (";
588         power_upgrade_string += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
589         power_upgrade_string += " K)\n";
590     }
591
592     else {
593         power_upgrade_string += " * MAX LEVEL * \n";
594     }
595
596     sf::Text power_upgrade_text = sf::Text(
597         power_upgrade_string,
598         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
599         16
600     );
601
602     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
603     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
604     power_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
605
606     this->render_window_ptr->draw(power_upgrade_text);
607
608
609     // 3. draw energy capacity (storage) upgrade sprite
610     this->render_window_ptr->draw(this->storage_upgrade_sprite);
611     this->render_window_ptr->draw(this->upgrade_plus_sprite);
612
613

```

```

614 // 4. draw energy capacity (storage) upgrade text
615 // 16 char line = " \n"
616 std::string energy_upgrade_string = "ENERGY CAPACITY \n";
617 energy_upgrade_string += " \n";
618
619 energy_upgrade_string += "CAPACITY: ";
620 energy_upgrade_string += std::to_string(this->storage_level * 200);
621 energy_upgrade_string += " kWh\n";
622
623 energy_upgrade_string += "LEVEL: ";
624 energy_upgrade_string += std::to_string(this->storage_level);
625 energy_upgrade_string += "\n\n";
626
627 if (this->storage_level < MAX_STORAGE_LEVELS) {
628     energy_upgrade_string += "[D]: + 200 kWh (";
629     energy_upgrade_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
630     energy_upgrade_string += " K)\n";
631 }
632
633 else {
634     energy_upgrade_string += " * MAX LEVEL * \n";
635 }
636
637 sf::Text energy_upgrade_text = sf::Text(
638     energy_upgrade_string,
639     *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
640     16
641 );
642
643 energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
644 energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
645 energy_upgrade_text.setFillColor(MONochrome_TEXT_GREEN);
646
647 this->render_window_ptr->draw(energy_upgrade_text);
648
649 return;
650 } /* __drawUpgradeOptions() */

```

4.14.3.8 __handleKeyPressEvents()

```

void WaveEnergyConverter::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

426 {
427     if (this->just_built) {
428         return;
429     }
430
431     switch (this->event_ptr->key.code) {
432         case (sf::Keyboard::U): {
433             this->__openUpgradeMenu();
434
435             break;
436         }
437
438         case (sf::Keyboard::W): {
439             if (this->production_menu_open) {
440                 this->dispatch_MWh++;
441
442                 if (this->dispatch_MWh > this->dispatchable_MWh) {
443                     this->dispatch_MWh = 0;
444                 }
445
446                 this->__computeProductionCosts();
447                 this->assets_manager_ptr->getSound("interface click")->play();
448             }
449
450             else if (this->upgrade_menu_open) {
451                 this->__upgradePowerCapacity();
452             }
453
454             break;
455         }
456
457         case (sf::Keyboard::S): {

```

```

460         if (this->production_menu_open) {
461             this->dispatch_MWh--;
462
463             if (this->dispatch_MWh < 0) {
464                 this->dispatch_MWh = this->dispatchable_MWh;
465             }
466
467             this->__computeProductionCosts();
468             this->assets_manager_ptr->getSound("interface click")->play();
469         }
470
471         break;
472     }
473
474
475     case (sf::Keyboard::D): {
476         if (this->upgrade_menu_open) {
477             this->__upgradeStorageCapacity();
478             this->__computeProduction();
479             this->__computeDispatch();
480         }
481
482         break;
483     }
484
485
486     default: {
487         // do nothing!
488
489         break;
490     }
491 }
492
493 return;
494 } /* __handleKeyPressEvents() */

```

4.14.3.9 __handleMouseButtonEvents()

```

void WaveEnergyConverter::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

509 {
510     if (this->just_built) {
511         return;
512     }
513     switch (this->event_ptr->mouseButton.button) {
514         case (sf::Mouse::Left): {
515             //...
516
517             break;
518         }
519
520
521         case (sf::Mouse::Right): {
522             //...
523
524             break;
525         }
526
527
528         default: {
529             // do nothing!
530
531             break;
532         }
533     }
534
535     return;
536 } /* __handleMouseButtonEvents() */

```


4.14.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
665 {
666     Message improvement_state_message;
667
668     improvement_state_message.channel = GAME_CHANNEL;
669     improvement_state_message.subject = "improvement state";
670
671     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
672     improvement_state_message.int_payload["operation_maintenance_cost"] =
673         this->operation_maintenance_cost;
674
675     this->message_hub_ptr->sendMessage(improvement_state_message);
676
677     std::cout << "Improvement state message sent by " << this << std::endl;
678
679     return;
680 } /* __sendImprovementStateMessage() */
```

4.14.3.11 __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() */
```

4.14.3.12 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

```
181 {
182     if (this->credits < WAVE_ENERGY_CONVERTER_BUILD_COST) {
183         std::cout << "Cannot upgrade wave energy converter: insufficient credits (need "
184             << WAVE_ENERGY_CONVERTER_BUILD_COST << " K)" << std::endl;
185
186         this->__sendInsufficientCreditsMessage();
187         return;
188     }
189
190     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
191         return;
192     }
193
194     this->health = 100;
195
196     this->capacity_kW += 100;
197     this->upgrade_level++;
198
199     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
200
201     this->__computeProduction();
202     this->__computeDispatch();
203
204     this->just_upgraded = true;
205
206     this->assets_manager_ptr->getSound("upgrade")->play();
207
208     this->__sendCreditsSpentMessage(WAVE_ENERGY_CONVERTER_BUILD_COST);
209     this->__sendTileStateRequest();
210     this->__sendGameStateRequest();
211
212     return;
213 } /* __upgradePowerCapacity() */
```

4.14.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from [TileImprovement](#).

```
885 {
886     // 1. update
887     this->__computeCapacityFactors();
888     this->update();
889
890     // 2. send improvement state message
891     this->__sendImprovementStateMessage();
892
893     // 3. handle start/stop
894     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
895         this->is_running = true;
896     }
897
898     else if (this->is_running and (this->dispatch_MWh <= 0)) {
899         this->is_running = false;
900     }
901
902     // 4. handle equipment health
903     if (this->is_running) {
904         this->health--;
905
906         if (this->health <= 0) {
907             this->__breakdown();
908         }
909     }
910 }
```

```

911 // 5. send tile state request (if selected)
912 if (this->is_selected) {
913     this->__sendTileStateRequest();
914 }
915
916 return;
917 } /* advanceTurn() */

```

4.14.3.14 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](#).

```

1006 {
1007     // 1. if just built, call base method and return
1008     if (this->just_built) {
1009         TileImprovement :: draw();
1010
1011         return;
1012     }
1013
1014     // 2. handle upgrade effects
1015     if (this->just_upgraded) {
1016         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1017             this->tile_improvement_sprite_animated[i].setColor(
1018                 sf::Color(
1019                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1020                     255,
1021                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1022                     255
1023                 )
1024             );
1025
1026             this->tile_improvement_sprite_animated[i].setScale(
1027                 sf::Vector2f(
1028                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1029                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1030                 )
1031             );
1032         }
1033
1034         this->upgrade_frame++;
1035     }
1036
1037     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1038         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1039             this->tile_improvement_sprite_animated[i].setColor(
1040                 sf::Color(255,255,255,255)
1041             );
1042
1043             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1044         }
1045
1046         this->just_upgraded = false;
1047         this->upgrade_frame = 0;
1048     }
1049
1050     // 3. draw first element of animated sprite
1051     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1052
1053     // 4. draw second element of animated sprite
1054     if (this->is_running) {
1055         this->tile_improvement_sprite_animated[0].setPosition(
1056             this->position_x,
1057             this->position_y + this->bobbing_y * cos(
1058                 (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1059             )
1060         );
1061
1062         this->tile_improvement_sprite_animated[1].setPosition(
1063             this->position_x,

```

```

1067         this->position_y + 1.25 * this->bobbing_y * sin(
1068             (double) (0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1069         )
1070     };
1071 }
1072
1073 else {
1074     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1075         this->tile_improvement_sprite_animated[i].setPosition(
1076             this->position_x,
1077             this->position_y + this->bobbing_y * cos(
1078                 (double) (0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1079             )
1080         );
1081     }
1082 }
1083
1084 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1085
1086 // 5. draw storage upgrades
1087 for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1088     this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1089 }
1090
1091 // 6. handle dispatch illustration
1092 if (this->dispatch_MWh > 0) {
1093     this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1094     this->__drawDispatch();
1095 }
1096
1097 // 7. draw production menu
1098 if (this->production_menu_open) {
1099     this->render_window_ptr->draw(this->production_menu_backing);
1100     this->render_window_ptr->draw(this->production_menu_backing_text);
1101
1102     this->__drawProductionMenu();
1103 }
1104
1105 // 8. draw upgrade menu
1106 if (this->upgrade_menu_open) {
1107     this->render_window_ptr->draw(this->upgrade_menu_backing);
1108     this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1109
1110     this->__drawUpgradeOptions();
1111 }
1112
1113 // 9. handle broken effects
1114 if (this->is_broken) {
1115     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1116         this->tile_improvement_sprite_animated[i].setColor(
1117             sf::Color(
1118                 255,
1119                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1120                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1121                 255
1122             )
1123         );
1124     }
1125 }
1126
1127 this->frame++;
1128 return;
1129 }
1130 }
1131
1132 /* draw() */

```

4.14.3.15 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

797 {
798     //          32 char x 17 line console "-----\n";
799     std::string options_substring = "CAPACITY: ";
800     options_substring += std::to_string(this->capacity_kW);
801     options_substring += " kW (level ";
802     options_substring += std::to_string(this->upgrade_level);
803     options_substring += ") \n";
804
805     options_substring += "PRODUCTION: ";
806     options_substring += std::to_string(this->production_MWh);
807     options_substring += " MWh\n";
808
809     options_substring += "DISPATCHABLE: ";
810     options_substring += std::to_string(this->dispatchable_MWh);
811     options_substring += " MWh\n";
812
813     options_substring += "HEALTH: ";
814     options_substring += std::to_string(this->health);
815     options_substring += "/100";
816
817     if (this->health <= 0) {
818         options_substring += " ** BROKEN! **\n";
819     }
820
821     else {
822         options_substring += "\n";
823     }
824
825     options_substring += " \n";
826     options_substring += " **** WAVE ENERGY OPTIONS **** \n";
827     options_substring += " \n";
828
829     options_substring += " [E]: ";
830
831     if (this->is_broken) {
832         options_substring += "*** BROKEN! ***\n";
833     }
834
835     else {
836         options_substring += "OPEN PRODUCTION MENU\n";
837     }
838
839     options_substring += " [U]: OPEN UPGRADE MENU \n";
840     options_substring += "HOLD [P]: SCRAP (";
841     options_substring += std::to_string(SCRAP_COST);
842     options_substring += " K)";
843
844     return options_substring;
845 } /* getTileOptionsSubstring() */

```

4.14.3.16 processEvent()

```

void WaveEnergyConverter::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

957 {
958     TileImprovement :: processEvent ();
959
960     if (this->event_ptr->type == sf::Event::KeyPressed) {
961         this->__handleKeyPressEvents();
962     }
963
964     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
965         this->__handleMouseButtonEvents();
966     }
967
968     return;
969 } /* processEvent() */

```

4.14.3.17 processMessage()

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

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

Reimplemented from [TileImprovement](#).

```
984 {
985     TileImprovement :: processMessage ();
986
987     //...
988
989     return;
990 } /* processMessage() */
```

4.14.3.18 setIsSelected()

```
void WaveEnergyConverter::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](#).

```
862 {
863     TileImprovement :: setIsSelected(is_selected);
864
865     if (this->is_running and this->is_selected) {
866         this->assets_manager_ptr->getSound("ocean waves")->play();
867     }
868
869     return;
870 } /* setIsSelected() */
```

4.14.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from [TileImprovement](#).

```
932 {
933     this->__computeProduction();
934     this->__computeProductionCosts();
935     this->__computeDispatch();
936
937     if (this->is_selected) {
938         this->__sendTileStateRequest();
939     }
940
941     return;
942 } /* update() */
```

4.14.4 Member Data Documentation

4.14.4.1 bobbing_y

```
double WaveEnergyConverter::bobbing_y
```

The bobbing extent of the wave energy converter.

4.14.4.2 capacity_factor_vec

```
std::vector<double> WaveEnergyConverter::capacity_factor_vec
```

A vector of daily capacity factors for the current month.

4.14.4.3 capacity_kW

```
int WaveEnergyConverter::capacity_kW
```

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

4.14.4.4 dispatch_MWh

```
int WaveEnergyConverter::dispatch_MWh
```

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

4.14.4.5 dispatch_vec_MWh

```
std::vector<double> WaveEnergyConverter::dispatch_vec_MWh
```

A vector of daily dispatch [MWh] for the current month.

4.14.4.6 dispatchable_MWh

```
int WaveEnergyConverter::dispatchable_MWh
```

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

4.14.4.7 max_daily_production_MWh

```
double WaveEnergyConverter::max_daily_production_MWh
```

The maximum daily production [MWh] of the solar PV array.

4.14.4.8 production_MWh

```
int WaveEnergyConverter::production_MWh
```

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

4.14.4.9 production_vec_MWh

```
std::vector<double> WaveEnergyConverter::production_vec_MWh
```

A vector of daily production [MWh] for the current month.

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

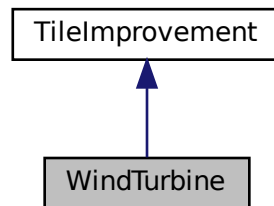
- header/[WaveEnergyConverter.h](#)
- source/[WaveEnergyConverter.cpp](#)

4.15 WindTurbine Class Reference

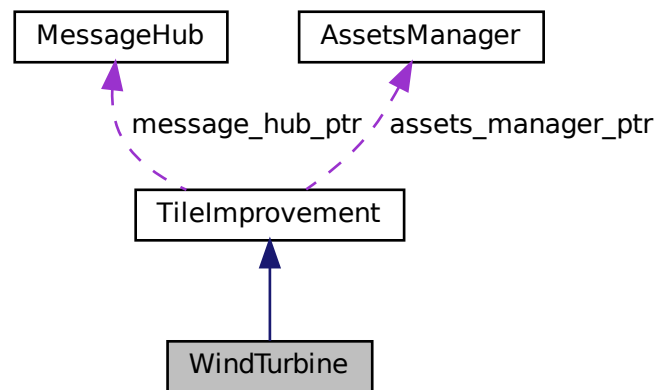
A settlement class (child class of [TileImprovement](#)).

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

Inheritance diagram for WindTurbine:



Collaboration diagram for WindTurbine:



Public Member Functions

- [WindTurbine](#) (double, double, int, 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 [setIsSelected](#) (bool)
Method to set the is selected attribute.
- void [advanceTurn](#) (void)

- *Method to handle turn advance.*
- void [update](#) (void)
- *Method to trigger production and dispatchable updates.*
- 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 [dispatch_MWh](#)
- *The current dispatch [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).*
- double [max_daily_production_MWh](#)
- *The maximum daily production [MWh] of the solar PV array.*
- double [rotor_drotation](#)
- *The rotation rate of the rotor.*
- std::vector< double > [capacity_factor_vec](#)
- *A vector of daily capacity factors for the current month.*
- std::vector< double > [production_vec_MWh](#)
- *A vector of daily production [MWh] for the current month.*
- std::vector< double > [dispatch_vec_MWh](#)
- *A vector of daily dispatch [MWh] for the current month.*

Private Member Functions

- void [__setUpTileImprovementSpriteAnimated](#) (void)
- *Helper method to set up tile improvement sprite (static).*
- void [__drawProductionMenu](#) (void)
- *Helper method to draw production menu assets.*
- void [__upgradePowerCapacity](#) (void)
- *Helper method to upgrade the power capacity.*
- void [__computeProductionCosts](#) (void)
- *Helper method to compute production costs (O&M) based on current production level.*
- void [__breakdown](#) (void)
- *Helper method to trigger an equipment breakdown.*
- void [__computeCapacityFactors](#) (void)
- *Helper method to compute capacity factors.*
- void [__computeProduction](#) (void)
- *Helper method to compute production values.*

- void [__computeDispatch](#) (void)
Helper method to compute dispatch values.
- 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.
- void [__sendImprovementStateMessage](#) (void)
Helper method to format and sent improvement state message.

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,
    int tile_resource,
    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>tile_resource</i>	The renewable resource quality 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.

```
735 :
736 TileImprovement (
```

```

737     position_x,
738     position_y,
739     tile_resource,
740     event_ptr,
741     render_window_ptr,
742     assets_manager_ptr,
743     message_hub_ptr
744 )
745 {
746     // 1. set attributes
747
748     // 1.1. private
749     //...
750
751     // 1.2. public
752     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
753
754     this->is_running = false;
755
756     this->health = 100;
757
758     this->capacity_kW = 100;
759     this->upgrade_level = 1;
760
761     this->storage_kWh = 0;
762     this->storage_level = 0;
763
764     this->production_MWh = 0;
765     this->dispatch_MWh = 0;
766     this->dispatchable_MWh = 0;
767
768     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
769
770     this->rotor_drotation = 256 * SECONDS_PER_FRAME;
771
772     this->capacity_factor_vec.resize(30, 0);
773     this->production_vec_MWh.resize(30, 0);
774     this->dispatch_vec_MWh.resize(30, 0);
775
776     this->tile_improvement_string = "WIND TURBINE";
777
778     this->__setUpTileImprovementSpriteAnimated();
779     this->__computeCapacityFactors();
780     this->update();
781
782     std::cout << "WindTurbine constructed at " << this << std::endl;
783
784     return;
785 } /* WindTurbine() */

```

4.15.2.2 ~WindTurbine()

```

WindTurbine::~WindTurbine (
    void ) [virtual]

```

Destructor for the [WindTurbine](#) class.

```

1131 {
1132     std::cout << "WindTurbine at " << this << " destroyed" << std::endl;
1133
1134     return;
1135 } /* ~WindTurbine() */

```

4.15.3 Member Function Documentation

4.15.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
250 {
251     TileImprovement :: __breakdown();
252
253     this->production_MWh = 0;
254     this->dispatch_MWh = 0;
255     this->dispatchable_MWh = 0;
256     this->operation_maintenance_cost = 0;
257
258     return;
259 } /* __breakdown() */
```

4.15.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
274 {
275     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
276     std::default_random_engine generator(seed);
277
278     double mean =
279         this->tile_resource_scalar * MEAN_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
280
281     double stdev = STDEV_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
282
283     if (this->tile_resource_scalar > 1) {
284         stdev /= this->tile_resource_scalar;
285     }
286
287     std::normal_distribution<double> normal_dist(mean, stdev);
288
289     double capacity_factor = 0;
290
291     for (int i = 0; i < 30; i++) {
292         capacity_factor = normal_dist(generator);
293
294         if (capacity_factor < 0) {
295             capacity_factor = 0;
296         }
297
298         this->capacity_factor_vec[i] = capacity_factor;
299     }
300
301     return;
302 } /* __computeCapacityFactors() */
```

4.15.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
345 {
346     double stored_energy_MWh = 0;
347     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
348
349     double demand_MWh = 0;
350     double production_MWh = 0;
351     double dispatchable_MWh = 0;
```

```

352     double difference_MWh = 0;
353
354     double room_MWh = 0;
355
356     for (int i = 0; i < 30; i++) {
357         demand_MWh = this->demand_vec_MWh[i];
358         production_MWh = this->production_vec_MWh[i];
359
360         if (production_MWh <= demand_MWh) {
361             this->dispatch_vec_MWh[i] = production_MWh;
362             dispatchable_MWh += this->dispatch_vec_MWh[i];
363
364             difference_MWh = demand_MWh - production_MWh;
365
366             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
367                 if (difference_MWh > stored_energy_MWh) {
368                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
369                     dispatchable_MWh += stored_energy_MWh;
370                     stored_energy_MWh = 0;
371                 }
372
373                 else {
374                     this->dispatch_vec_MWh[i] += difference_MWh;
375                     dispatchable_MWh += difference_MWh;
376                     stored_energy_MWh -= difference_MWh;
377                 }
378             }
379         }
380
381         else {
382             this->dispatch_vec_MWh[i] = demand_MWh;
383             dispatchable_MWh += this->dispatch_vec_MWh[i];
384
385             difference_MWh = production_MWh - demand_MWh;
386
387             if (
388                 (storage_capacity_MWh > 0) and
389                 (stored_energy_MWh < storage_capacity_MWh)
390             ) {
391                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
392
393                 if (difference_MWh > room_MWh) {
394                     stored_energy_MWh += room_MWh;
395                 }
396
397                 else {
398                     stored_energy_MWh += difference_MWh;
399                 }
400             }
401         }
402     }
403
404     this->dispatchable_MWh = round(dispatchable_MWh);
405
406     if (this->dispatch_MWh > this->dispatchable_MWh) {
407         this->dispatch_MWh = this->dispatchable_MWh;
408     }
409
410     return;
411 } /* __computeDispatch() */

```

4.15.3.4 __computeProduction()

```

void WindTurbine::__computeProduction (
    void ) [private]

```

Helper method to compute production values.

```

317 {
318     double production_MWh = 0;
319
320     for (int i = 0; i < 30; i++) {
321         this->production_vec_MWh[i] =
322             this->max_daily_production_MWh * this->capacity_factor_vec[i];
323
324         production_MWh += this->production_vec_MWh[i];
325     }
326
327     this->production_MWh = round(production_MWh);

```

```

328
329     return;
330 } /* __computeProduction() */

```

4.15.3.5 __computeProductionCosts()

```

void WindTurbine::__computeProductionCosts (
    void ) [private]

```

Helper method to compute production costs (O&M) based on current production level.

```

229 {
230     double operation_maintenance_cost =
231         (this->production_MWh * WIND_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
232     this->operation_maintenance_cost = round(operation_maintenance_cost);
233
234     return;
235 } /* __computeProductionCosts() */

```

4.15.3.6 __drawProductionMenu()

```

void WindTurbine::__drawProductionMenu (
    void ) [private]

```

Helper method to draw production menu assets.

```

114 {
115     // 1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
127         this->tile_improvement_sprite_animated[i].setRotation(0);
128
129         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
130
131         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
132         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
133         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
134         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
135     }
136
137     // 2. draw production text
138     std::string production_string = "[W]: INCREASE DISPATCH\n";
139     production_string += "[S]: DECREASE DISPATCH\n";
140     production_string += "\n";
141
142     production_string += "DISPATCH: ";
143     production_string += std::to_string(this->dispatch_MWh);
144     production_string += " MWh (MAX ";
145     production_string += std::to_string(this->dispatchable_MWh);
146     production_string += ")\n";
147
148     production_string += "O&M COST: ";
149     production_string += std::to_string(this->operation_maintenance_cost);
150     production_string += " K\n";
151
152     sf::Text production_text(
153         production_string,
154         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
155         16
156     );
157
158     production_text.setOrigin(production_text.getLocalBounds().width / 2, 0);

```

```

159     production_text.setFillColor(MONOCROME_TEXT_GREEN);
160
161     production_text.setPosition(400 + 30, 400 - 45);
162
163     this->render_window_ptr->draw(production_text);
164
165     return;
166 } /* __drawProductionMenu() */

```

4.15.3.7 __drawUpgradeOptions()

```

void WindTurbine::__drawUpgradeOptions (
    void ) [private]

```

Helper method to set up and draw upgrade options.

```

552 {
553     // 1. draw power capacity upgrade sprite
554     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
555         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
556         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
557
558         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
559         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
560
561         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
562         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
563
564         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
565         this->tile_improvement_sprite_animated[i].setRotation(0);
566
567         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
568
569         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
570         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
571         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
572         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
573     }
574
575     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
576
577
578     // 2. draw power capacity upgrade text
579     //          16 char line = "                \n"
580     std::string power_upgrade_string = "POWER CAPACITY \n";
581     power_upgrade_string += "                \n";
582
583     power_upgrade_string += "CAPACITY: ";
584     power_upgrade_string += std::to_string(this->capacity_kw);
585     power_upgrade_string += " kW\n";
586
587     power_upgrade_string += "LEVEL: ";
588     power_upgrade_string += std::to_string(this->upgrade_level);
589     power_upgrade_string += "\n\n";
590
591     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
592         power_upgrade_string += "[W]: + 100 kW (";
593         power_upgrade_string += std::to_string(WIND_TURBINE_BUILD_COST);
594         power_upgrade_string += " K)\n";
595     }
596
597     else {
598         power_upgrade_string += " * MAX LEVEL * \n";
599     }
600
601     sf::Text power_upgrade_text = sf::Text(
602         power_upgrade_string,
603         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
604         16
605     );
606
607     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
608     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
609     power_upgrade_text.setFillColor(MONOCROME_TEXT_GREEN);
610
611     this->render_window_ptr->draw(power_upgrade_text);
612
613
614     // 3. draw energy capacity (storage) upgrade sprite

```



```

615     this->render_window_ptr->draw(this->storage_upgrade_sprite);
616     this->render_window_ptr->draw(this->upgrade_plus_sprite);
617
618
619     // 4. draw energy capacity (storage) upgrade text
620     // 16 char line = " \n"
621     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
622     energy_upgrade_string += " \n";
623
624     energy_upgrade_string += "CAPACITY: ";
625     energy_upgrade_string += std::to_string(this->storage_level * 200);
626     energy_upgrade_string += " kWh\n";
627
628     energy_upgrade_string += "LEVEL: ";
629     energy_upgrade_string += std::to_string(this->storage_level);
630     energy_upgrade_string += "\n\n";
631
632     if (this->storage_level < MAX_STORAGE_LEVELS) {
633         energy_upgrade_string += "[D]: + 200 kWh (";
634         energy_upgrade_string += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
635         energy_upgrade_string += " K)\n";
636     }
637
638     else {
639         energy_upgrade_string += " * MAX LEVEL * \n";
640     }
641
642     sf::Text energy_upgrade_text = sf::Text(
643         energy_upgrade_string,
644         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
645         16
646     );
647
648     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
649     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
650     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
651
652     this->render_window_ptr->draw(energy_upgrade_text);
653
654     return;
655 } /* __drawUpgradeOptions() */

```

4.15.3.8 __handleKeyPressEvents()

```

void WindTurbine::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

426 {
427     if (this->just_built) {
428         return;
429     }
430
431     switch (this->event_ptr->key.code) {
432         case (sf::Keyboard::U): {
433             this->__openUpgradeMenu();
434
435             break;
436         }
437
438
439         case (sf::Keyboard::W): {
440             if (this->production_menu_open) {
441                 this->dispatch_MWh++;
442
443                 if (this->dispatch_MWh > this->dispatchable_MWh) {
444                     this->dispatch_MWh = 0;
445                 }
446
447                 this->__computeProductionCosts();
448                 this->assets_manager_ptr->getSound("interface click")->play();
449             }
450
451             else if (this->upgrade_menu_open) {
452                 this->__upgradePowerCapacity();
453             }
454
455             break;

```

```

456     }
457
458
459     case (sf::Keyboard::S): {
460         if (this->production_menu_open) {
461             this->dispatch_MWh--;
462
463             if (this->dispatch_MWh < 0) {
464                 this->dispatch_MWh = this->dispatchable_MWh;
465             }
466
467             this->__computeProductionCosts();
468             this->assets_manager_ptr->getSound("interface click")->play();
469         }
470
471         break;
472     }
473
474
475     case (sf::Keyboard::D): {
476         if (this->upgrade_menu_open) {
477             this->__upgradeStorageCapacity();
478             this->__computeProduction();
479             this->__computeDispatch();
480         }
481
482         break;
483     }
484
485
486     default: {
487         // do nothing!
488
489         break;
490     }
491 }
492
493 return;
494 } /* __handleKeyPressEvents() */

```

4.15.3.9 __handleMouseButtonEvents()

```

void WindTurbine::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

509 {
510     if (this->just_built) {
511         return;
512     }
513
514     switch (this->event_ptr->mouseButton.button) {
515         case (sf::Mouse::Left): {
516             //...
517
518             break;
519         }
520
521
522         case (sf::Mouse::Right): {
523             //...
524
525             break;
526         }
527
528
529         default: {
530             // do nothing!
531
532             break;
533         }
534     }
535
536     return;
537 } /* __handleMouseButtonEvents() */

```

4.15.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
670 {
671     Message improvement_state_message;
672
673     improvement_state_message.channel = GAME_CHANNEL;
674     improvement_state_message.subject = "improvement state";
675
676     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
677     improvement_state_message.int_payload["operation_maintenance_cost"] =
678         this->operation_maintenance_cost;
679
680     this->message_hub_ptr->sendMessage(improvement_state_message);
681
682     std::cout << "Improvement state message sent by " << this << std::endl;
683
684     return;
685 } /* __sendImprovementStateMessage() */
```

4.15.3.11 __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() */
```

4.15.3.12 __upgradePowerCapacity()

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

Helper method to upgrade the power capacity.

```
181 {
182     if (this->credits < WIND_TURBINE_BUILD_COST) {
183         std::cout << "Cannot upgrade wind turbine: insufficient credits (need "
184             << WIND_TURBINE_BUILD_COST << " K)" << std::endl;
185
186         this->__sendInsufficientCreditsMessage();
187         return;
188     }
189
190     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
191         return;
192     }
193
194     this->health = 100;
195
196     this->capacity_kW += 100;
197     this->upgrade_level++;
198
199     this->max_daily_production_MWh = (double) (24 * this->capacity_kW) / 1000;
200
201     this->__computeProduction();
202     this->__computeDispatch();
203
204     this->just_upgraded = true;
205
206     this->assets_manager_ptr->getSound("upgrade")->play();
207
208     this->__sendCreditsSpentMessage(WIND_TURBINE_BUILD_COST);
209     this->__sendTileStateRequest();
210     this->__sendGameStateRequest();
211
212     return;
213 } /* __upgradePowerCapacity() */
```

4.15.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from [TileImprovement](#).

```
890 {
891     // 1. update
892     this->__computeCapacityFactors();
893     this->update();
894
895     // 2. send improvement state message
896     this->__sendImprovementStateMessage();
897
898     // 3. handle start/stop
899     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
900         this->is_running = true;
901     }
902
903     else if (this->is_running and (this->dispatch_MWh <= 0)) {
904         this->is_running = false;
905     }
906
907     // 4. handle equipment health
908     if (this->is_running) {
909         this->health--;
910
911         if (this->health <= 0) {
912             this->__breakdown();
913         }
914     }
915 }
```

```

916 // 5. send tile state request (if selected)
917 if (this->is_selected) {
918     this->__sendTileStateRequest();
919 }
920
921 return;
922 } /* advanceTurn() */

```

4.15.3.14 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](#).

```

1011 {
1012 // 1. if just built, call base method and return
1013 if (this->just_built) {
1014     TileImprovement :: draw();
1015
1016     return;
1017 }
1018
1019 // 2. handle upgrade effects
1020 if (this->just_upgraded) {
1021     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1022         this->tile_improvement_sprite_animated[i].setColor(
1023             sf::Color(
1024                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1025                 255,
1026                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1027                 255
1028             )
1029         );
1030     }
1031
1032     this->tile_improvement_sprite_animated[i].setScale(
1033         sf::Vector2f(
1034             1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1035             1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1036         )
1037     );
1038 }
1039
1040 this->upgrade_frame++;
1041 }
1042
1043 if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1044     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1045         this->tile_improvement_sprite_animated[i].setColor(
1046             sf::Color(255,255,255,255)
1047         );
1048
1049         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1050     }
1051
1052     this->just_upgraded = false;
1053     this->upgrade_frame = 0;
1054 }
1055
1056 // 3. draw first element of animated sprite
1057 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1058
1059 // 4. draw second element of animated sprite
1060 if (this->is_running) {
1061     this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1062 }
1063
1064 this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1065
1066 // 5. draw storage upgrades
1067 for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1068     this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1069 }
1070
1071

```

```

1072     }
1073
1074
1075     // 6. handle dispatch illustration
1076     if (this->dispatch_MWh > 0) {
1077         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1078         this->__drawDispatch();
1079     }
1080
1081
1082     // 7. draw production menu
1083     if (this->production_menu_open) {
1084         this->render_window_ptr->draw(this->production_menu_backing);
1085         this->render_window_ptr->draw(this->production_menu_backing_text);
1086
1087         this->__drawProductionMenu();
1088     }
1089
1090
1091     // 8. draw upgrade menu
1092     if (this->upgrade_menu_open) {
1093         this->render_window_ptr->draw(this->upgrade_menu_backing);
1094         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1095
1096         this->__drawUpgradeOptions();
1097     }
1098
1099
1100     // 9. handle broken effects
1101     if (this->is_broken) {
1102         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1103             this->tile_improvement_sprite_animated[i].setColor(
1104                 sf::Color(
1105                     255,
1106                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1107                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1108                     255
1109                 )
1110             );
1111         }
1112     }
1113
1114     this->frame++;
1115     return;
1116 } /* draw() */

```

4.15.3.15 getTileOptionsSubstring()

```

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

```

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from [TileImprovement](#).

```

802 {
803     //          32 char x 17 line console "-----\n";
804     std::string options_substring = "CAPACITY: ";
805     options_substring += std::to_string(this->capacity_kW);
806     options_substring += " kW (level ";
807     options_substring += std::to_string(this->upgrade_level);
808     options_substring += ")\n";
809
810     options_substring += "PRODUCTION: ";
811     options_substring += std::to_string(this->production_MWh);
812     options_substring += " MWh\n";
813
814     options_substring += "DISPATCHABLE: ";
815     options_substring += std::to_string(this->dispatchable_MWh);
816     options_substring += " MWh\n";
817

```

```

818     options_substring           += "HEALTH:      ";
819     options_substring           += std::to_string(this->health);
820     options_substring           += "/100";
821
822     if (this->health <= 0) {
823         options_substring       += " ** BROKEN! **\n";
824     }
825
826     else {
827         options_substring       += "\n";
828     }
829
830     options_substring           += "
831     options_substring           += " **** WIND TURBINE OPTIONS ****
832     options_substring           += "
833
834     options_substring           += "      [E]:  ";
835
836     if (this->is_broken) {
837         options_substring       += " *** BROKEN! ***\n";
838     }
839
840     else {
841         options_substring       += "OPEN PRODUCTION MENU\n";
842     }
843
844     options_substring           += "      [U]:  OPEN UPGRADE MENU
845     options_substring           += "HOLD [P]:  SCRAP (";
846     options_substring           += std::to_string(SCRAP_COST);
847     options_substring           += " K)";
848
849     return options_substring;
850 } /* getTileOptionsSubstring() */

```

4.15.3.16 processEvent()

```

void WindTurbine::processEvent (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

962 {
963     TileImprovement :: processEvent ();
964
965     if (this->event_ptr->type == sf::Event::KeyPressed) {
966         this->__handleKeyPressEvents ();
967     }
968
969     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
970         this->__handleMouseButtonEvents ();
971     }
972
973     return;
974 } /* processEvent () */

```

4.15.3.17 processMessage()

```

void WindTurbine::processMessage (
    void ) [virtual]

```

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

Reimplemented from [TileImprovement](#).

```

989 {
990     TileImprovement :: processMessage ();
991
992     //...
993
994     return;
995 } /* processMessage () */

```

4.15.3.18 setIsSelected()

```
void WindTurbine::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](#).

```
867 {
868     TileImprovement :: setIsSelected(is_selected);
869
870     if (this->is_running and this->is_selected) {
871         this->assets_manager_ptr->getSound("wind turbine running")->play();
872     }
873
874     return;
875 } /* setIsSelected() */
```

4.15.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from [TileImprovement](#).

```
937 {
938     this->__computeProduction();
939     this->__computeProductionCosts();
940     this->__computeDispatch();
941
942     if (this->is_selected) {
943         this->__sendTileStateRequest();
944     }
945
946     return;
947 } /* update() */
```

4.15.4 Member Data Documentation

4.15.4.1 capacity_factor_vec

```
std::vector<double> WindTurbine::capacity_factor_vec
```

A vector of daily capacity factors for the current month.

4.15.4.2 capacity_kW

```
int WindTurbine::capacity_kW
```

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

4.15.4.3 dispatch_MWh

```
int WindTurbine::dispatch_MWh
```

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

4.15.4.4 dispatch_vec_MWh

```
std::vector<double> WindTurbine::dispatch_vec_MWh
```

A vector of daily dispatch [MWh] for the current month.

4.15.4.5 dispatchable_MWh

```
int WindTurbine::dispatchable_MWh
```

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

4.15.4.6 max_daily_production_MWh

```
double WindTurbine::max_daily_production_MWh
```

The maximum daily production [MWh] of the solar PV array.

4.15.4.7 production_MWh

```
int WindTurbine::production_MWh
```

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

4.15.4.8 production_vec_MWh

```
std::vector<double> WindTurbine::production_vec_MWh
```

A vector of daily production [MWh] for the current month.

4.15.4.9 rotor_drotation

```
double WindTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

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

Chapter 5

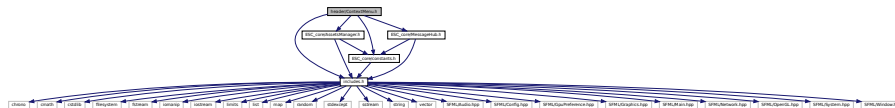
File Documentation

5.1 header/ContextMenu.h File Reference

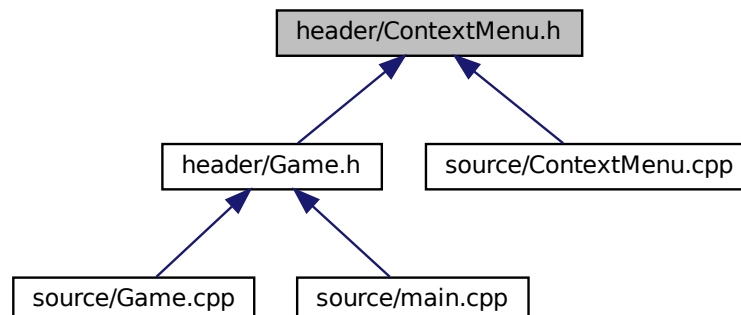
Header file for the [ContextMenu](#) class.

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

Include dependency graph for ContextMenu.h:



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



Classes

- class [ContextMenu](#)

A class which defines a context menu for the game.

Enumerations

- enum [ConsoleState](#) {
[NONE_STATE](#) , [READY](#) , [MENU](#) , [TILE](#) ,
[N_CONSOLE_STATES](#) }

An enumeration of the different console screen states.

5.1.1 Detailed Description

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

5.1.2 Enumeration Type Documentation

5.1.2.1 ConsoleState

enum [ConsoleState](#)

An enumeration of the different console screen states.

Enumerator

NONE_STATE	None state (for initialization)
READY	Ready (default) state.
MENU	Game menu state.
TILE	Tile context state.
N_CONSOLE_STATES	A simple hack to get the number of console screen states.

```

68         {
69     NONE\_STATE,
70     READY,
71     MENU,
72     TILE,
73     N\_CONSOLE\_STATES
74 };

```

5.2 header/DieselGenerator.h File Reference

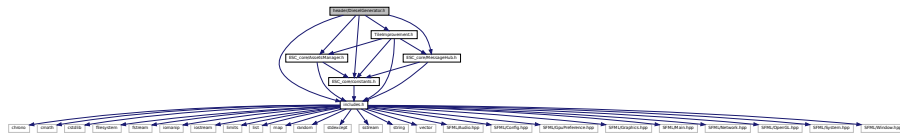
Header file for the [DieselGenerator](#) class.

```

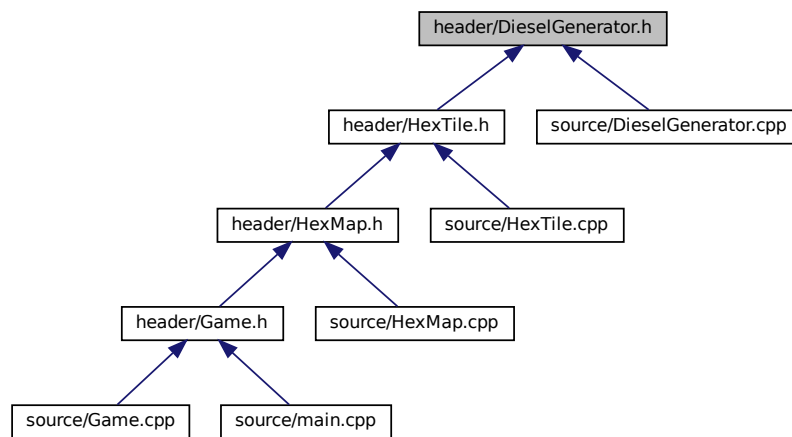
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"

```

```
#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

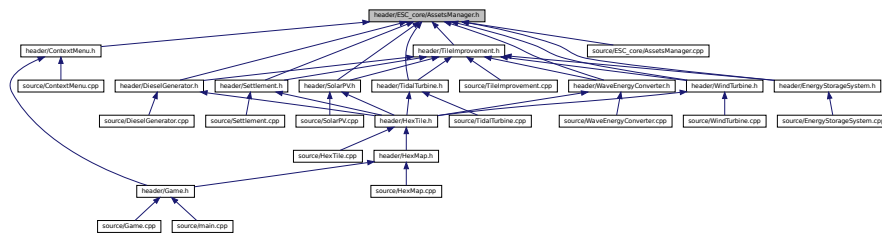
Header file for the [DieselGenerator](#) class.

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

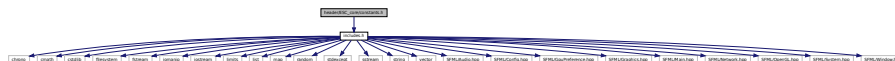
Header file for the [AssetsManager](#) class.

5.5 header/ESC_core/constants.h File Reference

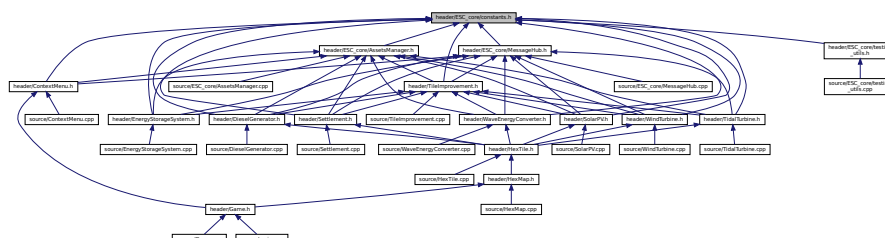
Header file for various constants.

```
#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 std::string `SETTLEMENT_CHANNEL` = "SETTLEMENT CHANNEL"
- A message channel for the settlement.*

 - 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` = 999999
- The number of credits (x1000) earned.*

 - const double `CREDITS_PER_MWH_SERVED` = 1
- The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.*

 - const int `EMISSIONS_LIFETIME_LIMIT_TONNES` = 1500
- The cost of doing a resource assessment.*

 - const int `RESOURCE_ASSESSMENT_COST` = 20
- The cost of building a settlement.*

 - const int `BUILD_SETTLEMENT_COST` = 250
- The starting population of a settlement.*

 - const int `STARTING_POPULATION` = 100
- The monthly population growth rate.*

 - const double `POPULATION_MONTHLY_GROWTH_RATE` = 1.005
- The litres of diesel consumed in producing 1 MWh (assumes higher heating value and constant thermal efficiency of 0.25).*

 - const double `LITRES_DIESEL_PER_MWH_PRODUCTION` = 373.175
- The cost of a litre of diesel.*

 - const double `COST_PER_LITRE_DIESEL` = 1.70

- const double `KG_CO2E_PER_LITRE_DIESEL` = 3.1596
The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.
- const double `DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION` = 50
The operation and maintenace cost of running a diesel generator (assumed 0.05 credits per kWh produced).
- const double `SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION` = 10
The operation and maintenance cost of running a solar PV array (assumed 0.01 credits per kWh produced).
- const double `TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION` = 50
The operation and maintenance cost of running a tidal turbine (assumed 0.05 credits per kWh produced).
- const double `WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION` = 50
The operation and maintenance cost of running a wave energy converter (assumed 0.05 credits per kWh produced).
- const double `WIND_OP_MAINT_COST_PER_MWH_PRODUCTION` = 50
The operation and maintenance cost of running a wind turbine (assumed 0.05 credits per kWh produced).
- const std::vector< double > `MEAN_DAILY_DEMAND_RATIOS`
The mean daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const std::vector< double > `STDEV_DAILY_DEMAND_RATIOS`
The standard deviation in daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const double `MAXIMUM_DAILY_DEMAND_PER_CAPITA` = 0.0475
The maximum daily demand [MWh] (at any point in the year) per capita.
- const std::vector< double > `MEAN_DAILY_SOLAR_CAPACITY_FACTORS`
The mean daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > `STDEV_DAILY_SOLAR_CAPACITY_FACTORS`
The standard deviation in daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const double `DAILY_TIDAL_CAPACITY_FACTOR` = 0.2175
The daily tidal capacity factor, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000. The tides are not a random process, and are not very sensitive to season.
- const std::vector< double > `MEAN_DAILY_WAVE_CAPACITY_FACTORS`
The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > `STDEV_DAILY_WAVE_CAPACITY_FACTORS`
The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > `MEAN_DAILY_WIND_CAPACITY_FACTORS`
The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > `STDEV_DAILY_WIND_CAPACITY_FACTORS`
The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- 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 COST_PER_LITRE_DIESEL

```
const double COST_PER_LITRE_DIESEL = 1.70
```

The cost of a litre of diesel.

5.5.3.6 CREDITS_PER_MWH_SERVED

```
const double CREDITS_PER_MWH_SERVED = 1
```

The number of credits (x1000) earned.

5.5.3.7 DAILY_TIDAL_CAPACITY_FACTOR

```
const double DAILY_TIDAL_CAPACITY_FACTOR = 0.2175
```

The daily tidal capacity factor, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000. The tides are not a random process, and are not very sensitive to season.

5.5.3.8 DIESEL_GENERATOR_BUILD_COST

```
const int DIESEL_GENERATOR_BUILD_COST = 100
```

The cost of building (or upgrading) a diesel generator in 100 kW increments.

5.5.3.9 DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenace cost of running a diesel generator (assumed 0.05 credits per kWh produced).

5.5.3.10 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.

5.5.3.11 ENERGY_STORAGE_SYSTEM_BUILD_COST

```
const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 160
```

The cost of adding energy storage in 200 kWh increments.

5.5.3.12 FLOAT_TOLERANCE

```
const double FLOAT_TOLERANCE = 1e-6
```

Tolerance for floating point equality tests.

5.5.3.13 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

5.5.3.14 GAME_CHANNEL

```
const std::string GAME_CHANNEL = "GAME CHANNEL"
```

A message channel for game messages.

5.5.3.15 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

5.5.3.16 GAME_STATE_CHANNEL

```
const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"
```

A message channel for game state messages.

5.5.3.17 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.18 HEX_MAP_CHANNEL

```
const std::string HEX_MAP_CHANNEL = "HEX MAP CHANNEL"
```

A message channel for hex map messages.

5.5.3.19 KG_CO2E_PER_LITRE_DIESEL

```
const double KG_CO2E_PER_LITRE_DIESEL = 3.1596
```

The CO₂-equivalent mass of emissions that result from burning one litre of diesel fuel.

5.5.3.20 LITRES_DIESEL_PER_MWH_PRODUCTION

```
const double LITRES_DIESEL_PER_MWH_PRODUCTION = 373.175
```

The litres of diesel consumed in producing 1 MWh (assumes higher heating value and constant thermal efficiency of 0.25).

5.5.3.21 MAX_STORAGE_LEVELS

```
const int MAX_STORAGE_LEVELS = 5
```

The maximum storage level of any tile improvement.

5.5.3.22 MAX_UPGRADE_LEVELS

```
const int MAX_UPGRADE_LEVELS = 5
```

The maximum upgrade level of any tile improvement.

5.5.3.23 MAXIMUM_DAILY_DEMAND_PER_CAPITA

```
const double MAXIMUM_DAILY_DEMAND_PER_CAPITA = 0.0475
```

The maximum daily demand [MWh] (at any point in the year) per capita.

5.5.3.24 MEAN_DAILY_DEMAND_RATIOS

```
const std::vector<double> MEAN_DAILY_DEMAND_RATIOS
```

Initial value:

```
= {  
    0.702, 0.704, 0.652,  
    0.546, 0.445, 0.362,  
    0.261, 0.261, 0.379,  
    0.518, 0.622, 0.716  
}
```

The mean daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.

5.5.3.25 MEAN_DAILY_SOLAR_CAPACITY_FACTORS

```
const std::vector<double> MEAN_DAILY_SOLAR_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.022, 0.046, 0.088,  
    0.138, 0.171, 0.175,  
    0.164, 0.139, 0.104,  
    0.061, 0.030, 0.016  
}
```

The mean daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.26 MEAN_DAILY_WAVE_CAPACITY_FACTORS

```
const std::vector<double> MEAN_DAILY_WAVE_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.742, 0.694, 0.618,  
    0.467, 0.366, 0.292,  
    0.280, 0.293, 0.374,  
    0.424, 0.662, 0.600  
}
```

The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.27 MEAN_DAILY_WIND_CAPACITY_FACTORS

```
const std::vector<double> MEAN_DAILY_WIND_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.591, 0.594, 0.627,  
    0.629, 0.579, 0.537,  
    0.442, 0.507, 0.587,  
    0.618, 0.611, 0.580  
}
```

The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.28 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

5.5.3.29 POPULATION_MONTHLY_GROWTH_RATE

```
const double POPULATION_MONTHLY_GROWTH_RATE = 1.005
```

The monthly population growth rate.

5.5.3.30 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.31 SCRAP_COST

```
const int SCRAP_COST = 50
```

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

5.5.3.32 SECONDS_PER_FRAME

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

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

5.5.3.33 SECONDS_PER_MONTH

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

5.5.3.34 SECONDS_PER_YEAR

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

5.5.3.35 SETTLEMENT_CHANNEL

```
const std::string SETTLEMENT_CHANNEL = "SETTLEMENT CHANNEL"
```

A message channel for the settlement.

5.5.3.36 SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION = 10
```

The operation and maintenance cost of running a solar PV array (assumed 0.01 credits per kWh produced).

5.5.3.37 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.

5.5.3.38 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.39 STARTING_CREDITS

```
const int STARTING_CREDITS = 999999
```

5.5.3.40 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.41 STDEV_DAILY_DEMAND_RATIOS

```
const std::vector<double> STDEV_DAILY_DEMAND_RATIOS
```

Initial value:

```
= {  
    0.069, 0.074, 0.072,  
    0.072, 0.063, 0.060,  
    0.012, 0.031, 0.040,  
    0.049, 0.063, 0.053  
}
```

The standard deviation in daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.

5.5.3.42 STDEV_DAILY_SOLAR_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_SOLAR_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.013, 0.024, 0.043,  
    0.049, 0.072, 0.072,  
    0.076, 0.065, 0.048,  
    0.026, 0.018, 0.009  
}
```

The standard deviation in daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.43 STDEV_DAILY_WAVE_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_WAVE_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.146, 0.135, 0.163,  
    0.145, 0.158, 0.106,  
    0.086, 0.058, 0.145,  
    0.171, 0.184, 0.309  
}
```

The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.44 STDEV_DAILY_WIND_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_WIND_CAPACITY_FACTORS
```

Initial value:

```
= {  
    0.147, 0.142, 0.198,  
    0.154, 0.162, 0.202,  
    0.180, 0.217, 0.198,  
    0.168, 0.141, 0.168  
}
```

The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.45 TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a tidal turbine (assumed 0.05 credits per kWh produced).

5.5.3.46 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

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

5.5.3.47 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).

5.5.3.48 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.49 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.50 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).

5.5.3.51 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.

5.5.3.52 WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a wave energy converter (assumed 0.05 credits per kWh produced).

5.5.3.53 WIND_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double WIND_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a wind turbine (assumed 0.05 credits per kWh produced).

5.5.3.54 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

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

5.5.3.55 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

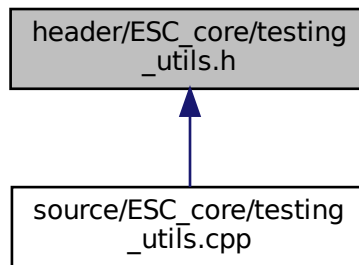
Header file which simply cites the doxygen tool.

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

```

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

```

```

188
189     throw std::runtime_error(error_str);
190     return;
191 } /* testFloatEquals() */

```

5.9.2.6 testGreaterThan()

```

void testGreaterThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x > y$.

Parameters

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

```

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

```

5.9.2.7 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if $x \geq y$.

Parameters

<i>x</i>	The first of two numbers to test.
----------	-----------------------------------

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() */

```

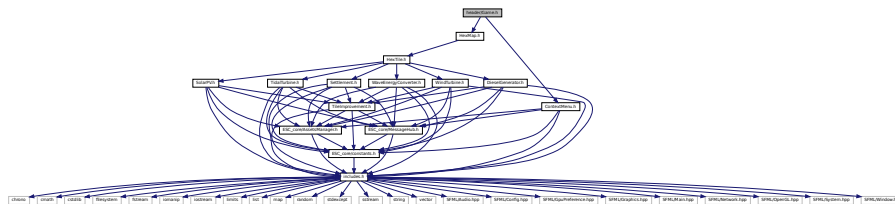
5.10 header/Game.h File Reference

```

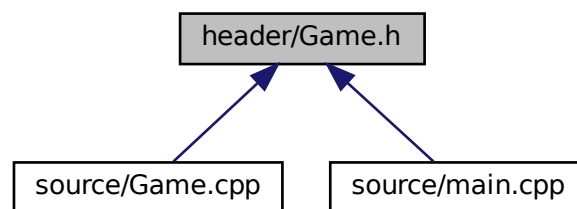
#include "HexMap.h"
#include "ContextMenu.h"

```

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 */

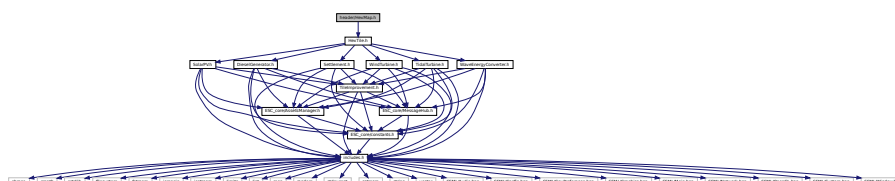
```

5.11 header/HexMap.h File Reference

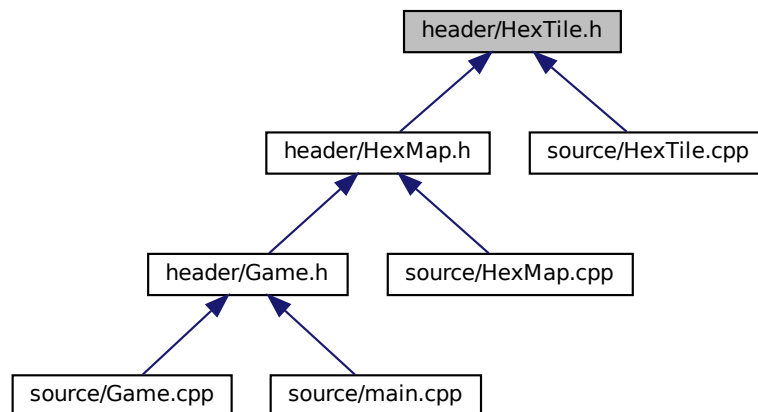
Header file for the [HexMap](#) class.

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

Include dependency graph for HexMap.h:



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



Classes

- class [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.

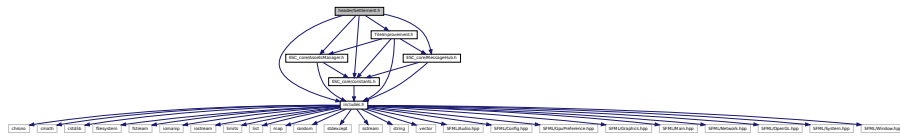
```

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

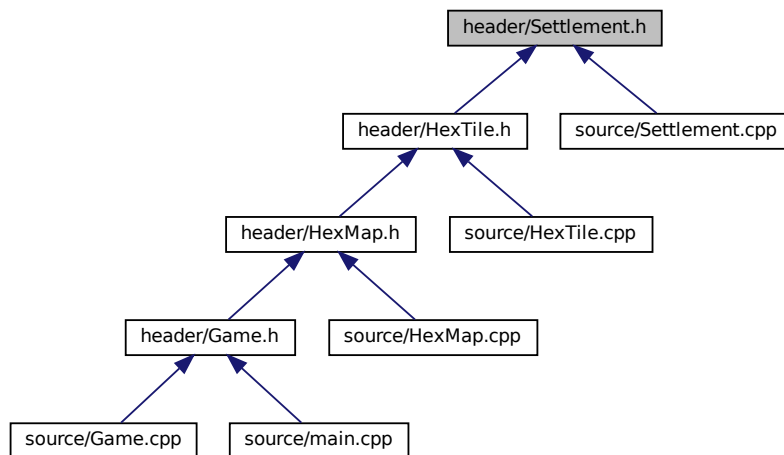
```

```
#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

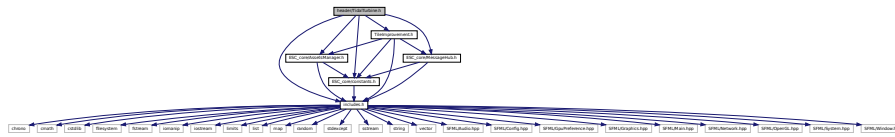
Header file for the [SolarPV](#) class.

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

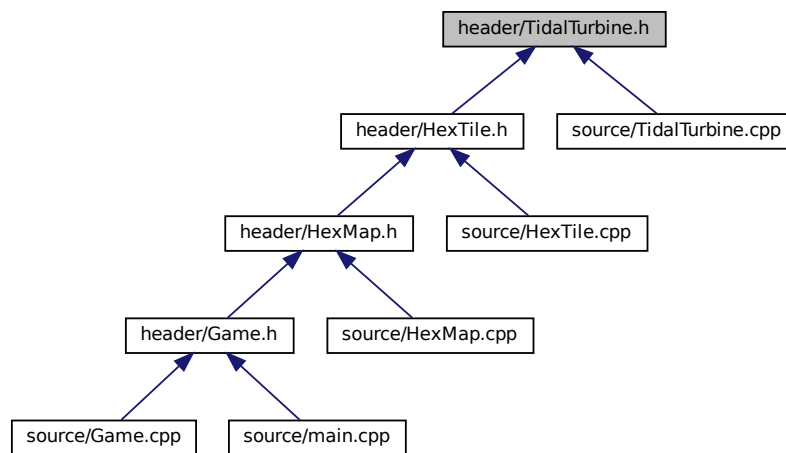


```
#include "TileImprovement.h"
```

Include dependency graph for 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"
```


Enumerator

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

```

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

```

5.17 header/WaveEnergyConverter.h File Reference

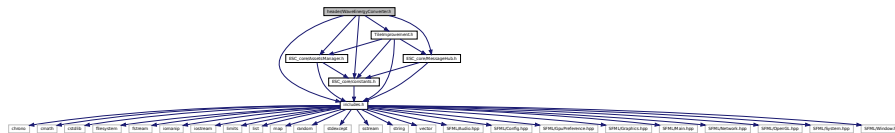
Header file for the [WaveEnergyConverter](#) class.

```

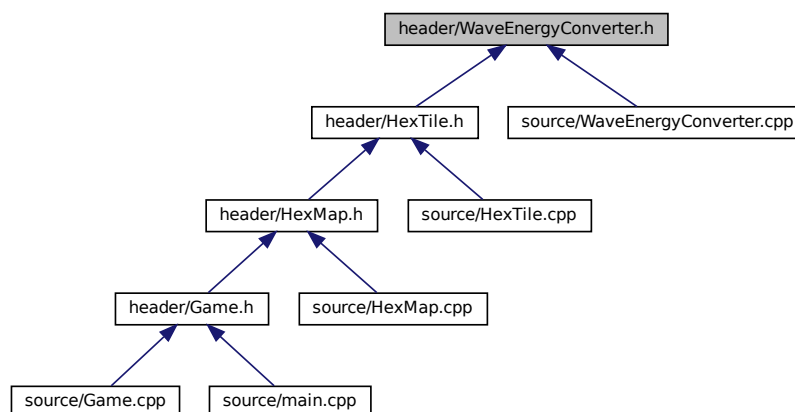
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"

```

Include dependency graph for WaveEnergyConverter.h:



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



Classes

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

5.17.1 Detailed Description

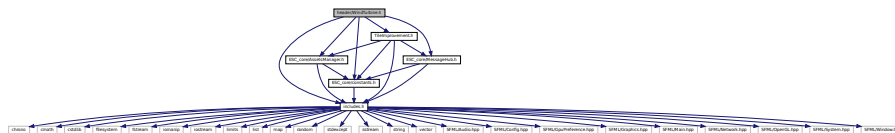
Header file for the [WaveEnergyConverter](#) class.

5.18 header/WindTurbine.h File Reference

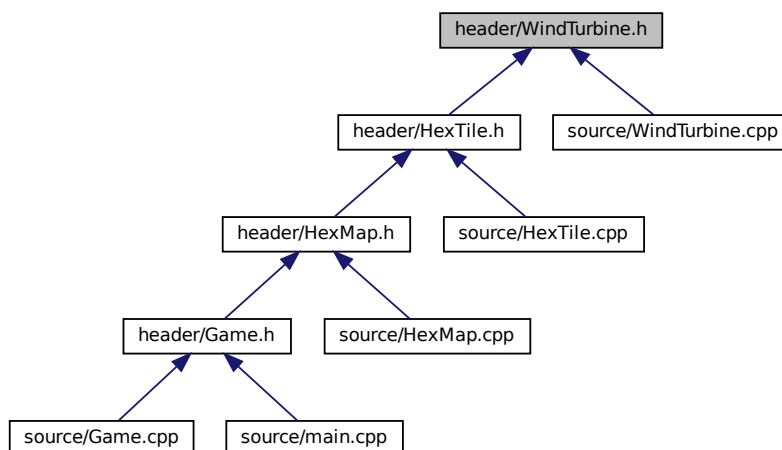
Header file for the [WindTurbine](#) class.

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

Include dependency graph for WindTurbine.h:



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



Classes

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

5.18.1 Detailed Description

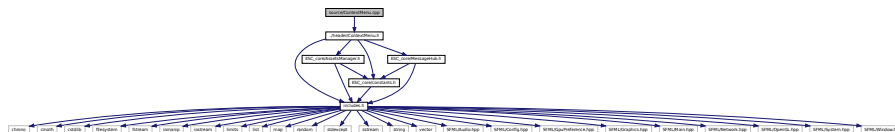
Header file for the `WindTurbine` class.

5.19 source/ContextMenu.cpp File Reference

Implementation file for the `ContextMenu` class.

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

Include dependency graph for ContextMenu.cpp:



5.19.1 Detailed Description

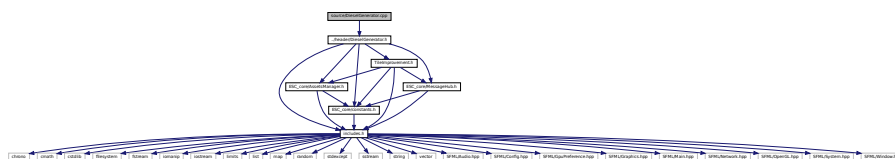
Implementation file for the `ContextMenu` class.

A class which defines a context menu for the game.

5.20 source/DieselGenerator.cpp File Reference

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

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



5.20.1 Detailed Description

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

A base class for the tile improvement hierarchy.

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 std::cout.
------------------	---

```

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 "__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() */

```

5.24.2.6 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\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() */

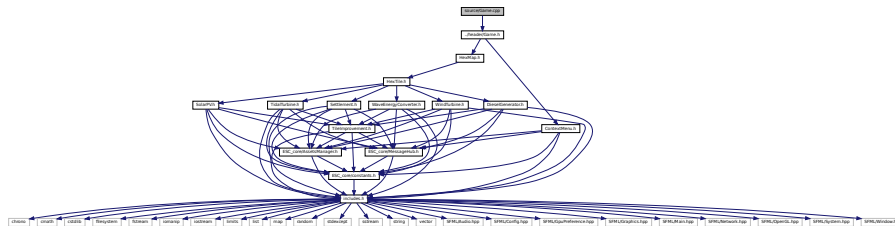
```

5.25 source/Game.cpp File Reference

Implementation file for the `Game` class.

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

Include dependency graph for Game.cpp:



5.25.1 Detailed Description

Implementation file for the `Game` class.

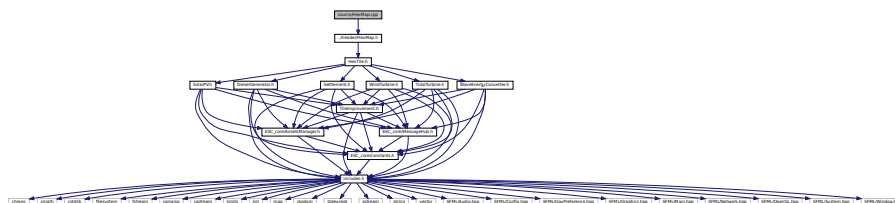
A class which defines a tile of a hex map.

5.26 source/HexMap.cpp File Reference

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

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

Include dependency graph for HexMap.cpp:



5.28.1 Detailed Description

Implementation file for `main()` for Road To Zero.

5.28.2 Function Documentation

5.28.2.1 `constructRenderWindow()`

```
sf::RenderWindow * constructRenderWindow (
    void )
```

Helper function to construct render window.

Returns

Pointer to the render window.

```
329 {
330     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
331         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
332         "Road To Zero"
333     );
334
335     return render_window_ptr;
336 } /* constructRenderWindow() */
```

5.28.2.2 `loadAssets()`

```
void loadAssets (
    AssetsManager * assets_manager_ptr )
```

Helper function to load game assets.

Parameters

<code>assets_manager_ptr</code>	Pointer to the assets manager.
---------------------------------	--------------------------------

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

```
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 assets_manager_ptr->loadTexture(
164     "assets/tile_sheets/coin_16x16_1_CC-BY.png",
165     "coin"
166 );
167
168
169 // 3. load sounds
170 assets_manager_ptr->loadSound(
171     "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
```

```
172     "coin ring"
173 );
174
175 assets_manager_ptr->loadSound(
176     "assets/audio/samples/mixkit-positive-notification-951_MixkitFree.ogg",
177     "positive notification"
178 );
179
180 assets_manager_ptr->loadSound(
181     "assets/audio/samples/mixkit-sci-fi-click-900_MixkitFree.ogg",
182     "sci-fi click"
183 );
184
185 assets_manager_ptr->loadSound(
186     "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932_MixkitFree.ogg",
187     "insufficient credits"
188 );
189
190 assets_manager_ptr->loadSound(
191     "assets/audio/samples/mixkit-data-scanner-2487_MixkitFree.ogg",
192     "resource assessment"
193 );
194
195 assets_manager_ptr->loadSound(
196     "assets/audio/samples/mixkit-interface-click-1126_MixkitFree.ogg",
197     "console string print"
198 );
199
200 assets_manager_ptr->loadSound(
201     "assets/audio/samples/mixkit-video-game-retro-click-237_MixkitFree.ogg",
202     "resource overlay toggle on"
203 );
204
205 assets_manager_ptr->loadSound(
206     "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED_MixkitFree.ogg",
207     "resource overlay toggle off"
208 );
209
210 assets_manager_ptr->loadSound(
211     "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703_MixkitFree.ogg",
212     "clear mountains tile"
213 );
214
215 assets_manager_ptr->loadSound(
216     "assets/audio/samples/mixkit-arcade-game-explosion-2759_MixkitFree.ogg",
217     "clear non-mountains tile"
218 );
219
220 assets_manager_ptr->loadSound(
221     "assets/audio/samples/mixkit-electronic-retro-block-hit-2185_MixkitFree.ogg",
222     "place improvement"
223 );
224
225 assets_manager_ptr->loadSound(
226     "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED_MixkitFree.ogg",
227     "build menu open"
228 );
229
230 assets_manager_ptr->loadSound(
231     "assets/audio/samples/mixkit-video-game-lock-2851_MixkitFree.ogg",
232     "build menu close"
233 );
234
235 assets_manager_ptr->loadSound(
236     "assets/audio/samples/mixkit-jump-into-the-water-1180_MixkitFree.ogg",
237     "splash"
238 );
239
240 assets_manager_ptr->loadSound(
241     "assets/audio/samples/505316__nuncaconoci__diesel_CC0.ogg",
242     "diesel running"
243 );
244
245 assets_manager_ptr->loadSound(
246     "assets/audio/samples/33460__pempi__320d_2_CC-BY.ogg",
247     "diesel start"
248 );
249
250 assets_manager_ptr->loadSound(
251     "assets/audio/samples/132724__andy_gardner__wind-turbine-blades_CC-BY.ogg",
252     "wind turbine running"
253 );
254
255 assets_manager_ptr->loadSound(
256     "assets/audio/samples/58416__darren1979__oceanwaves_CC-SAMPLING.ogg",
257     "ocean waves"
258 );
```

```

259
260     assets_manager_ptr->loadSound(
261         "assets/audio/samples/369927__mephisto_egmont__water-flowing-in-tubes_CC-BY.ogg",
262         "water flow"
263     );
264
265     assets_manager_ptr->loadSound(
266         "assets/audio/samples/647663__jotraing__electric-train-motor-idle-loop-new-generation-rollingstock_CC0.ogg",
267         "solar hum"
268     );
269
270     assets_manager_ptr->loadSound(
271         "assets/audio/samples/mixkit-epic-futuristic-movie-accent-2913_MixkitFree.ogg",
272         "game title screen"
273     );
274
275     assets_manager_ptr->loadSound(
276         "assets/audio/samples/mixkit-calm-park-with-people-and-children_MixkitFree.ogg",
277         "people and children"
278     );
279
280     assets_manager_ptr->loadSound(
281         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
282         "upgrade"
283     );
284
285     assets_manager_ptr->loadSound(
286         "assets/audio/samples/mixkit-cool-interface-click-tone-2568_MixkitFree.ogg",
287         "interface click"
288     );
289
290     assets_manager_ptr->loadSound(
291         "assets/audio/samples/mixkit-factory-metal-hard-hit-2980_MixkitFree.ogg",
292         "breakdown"
293     );
294
295
296     // 4. load tracks
297     assets_manager_ptr->loadTrack(
298         "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
299         "Tree Star Moon - Dobranoc"
300     );
301
302     assets_manager_ptr->loadTrack(
303         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
304         "Tree Star Moon - Lighthouse"
305     );
306
307     assets_manager_ptr->loadTrack(
308         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
309         "Tree Star Moon - Sky Farm"
310     );
311
312     return;
313 } /* loadAssets() */

```

5.28.2.3 main()

```

int main (
    int argc,
    char ** argv )
{
    // 1. load assets
    AssetsManager assets_manager;
    loadAssets(&assets_manager);

    // 2. construct render window
    sf::RenderWindow* render_window_ptr = constructRenderWindow();

    // 3. start game loop
    bool quit_game = false;
    assets_manager.playTrack();

    while (not quit_game) {
        Game game(render_window_ptr, &assets_manager);
        quit_game = game.run();
    }
}

```



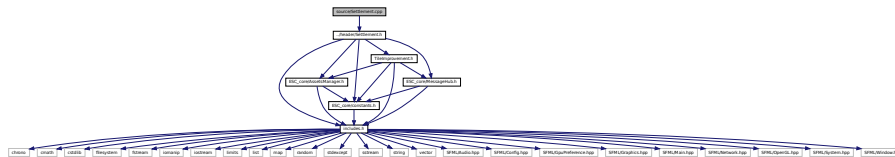
```
362 // 4. clean up
363 render_window_ptr->close();
364 delete render_window_ptr;
365
366 return 0;
367 } /* main() */
```

5.29 source/Settlement.cpp File Reference

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

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

Include dependency graph for Settlement.cpp:



5.29.1 Detailed Description

Implementation file for the **Settlement** class.

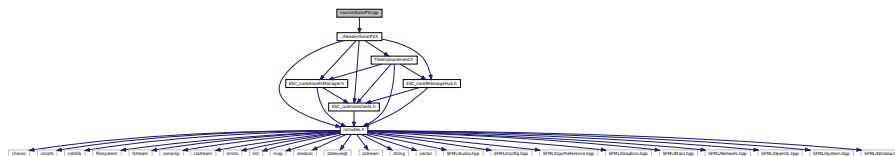
A base class for the tile improvement hierarchy.

5.30 source/SolarPV.cpp File Reference

Implementation file for the **SolarPV** class.

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

Include dependency graph for SolarPV.cpp:



5.30.1 Detailed Description

Implementation file for the **SolarPV** 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/>. 282

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 281

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 40, 53, 106, 159, 169, 186, 205, 227, 245

Index

- __advanceTurn
 - Game, [64](#)
- __assembleHexMap
 - HexMap, [83](#)
- __assessNeighbours
 - HexMap, [83](#)
- __breakdown
 - DieselGenerator, [41](#)
 - SolarPV, [170](#)
 - TidalTurbine, [187](#)
 - TileImprovement, [207](#)
 - WaveEnergyConverter, [228](#)
 - WindTurbine, [246](#)
- __buildDieselGenerator
 - HexTile, [108](#)
- __buildDrawOrderVector
 - HexMap, [84](#)
- __buildEnergyStorage
 - HexTile, [108](#)
- __buildSettlement
 - HexTile, [109](#)
- __buildSolarPV
 - HexTile, [109](#)
- __buildTidalTurbine
 - HexTile, [110](#)
- __buildWaveEnergyConverter
 - HexTile, [110](#)
- __buildWindTurbine
 - HexTile, [111](#)
- __checkTerminatingConditions
 - Game, [64](#)
- __clearDecoration
 - HexTile, [112](#)
- __closeBuildMenu
 - HexTile, [112](#)
- __closeProductionMenu
 - TileImprovement, [207](#)
- __closeUpgradeMenu
 - TileImprovement, [207](#)
- __computeCapacityFactors
 - SolarPV, [171](#)
 - TidalTurbine, [188](#)
 - WaveEnergyConverter, [229](#)
 - WindTurbine, [247](#)
- __computeCurrentDemand
 - Game, [64](#)
- __computeDispatch
 - SolarPV, [171](#)
 - TidalTurbine, [188](#)
- __computeProduction
 - WaveEnergyConverter, [229](#)
 - WindTurbine, [247](#)
- __computeProductionCosts
 - DieselGenerator, [41](#)
 - SolarPV, [172](#)
 - TidalTurbine, [189](#)
 - WaveEnergyConverter, [230](#)
 - WindTurbine, [248](#)
- __draw
 - Game, [65](#)
- __drawConsoleScreenFrame
 - ContextMenu, [22](#)
- __drawConsoleText
 - ContextMenu, [23](#)
- __drawDispatch
 - TileImprovement, [207](#)
- __drawFrameClockOverlay
 - Game, [65](#)
- __drawHUD
 - Game, [66](#)
- __drawProductionMenu
 - DieselGenerator, [41](#)
 - SolarPV, [173](#)
 - TidalTurbine, [189](#)
 - WaveEnergyConverter, [231](#)
 - WindTurbine, [249](#)
- __drawUpgradeOptions
 - SolarPV, [173](#)
 - TidalTurbine, [190](#)
 - WaveEnergyConverter, [232](#)
 - WindTurbine, [250](#)
- __drawVisualScreenFrame
 - ContextMenu, [24](#)
- __enforceOceanContinuity
 - HexMap, [84](#)
- __getMajorityTileType
 - HexMap, [85](#)
- __getNeighboursVector
 - HexMap, [86](#)
- __getNoise
 - HexMap, [87](#)
- __getSelectedTile
 - HexMap, [88](#)
- __getTileCoordsSubstring

- HexTile, 112
- __getTileImprovementSubstring
 - HexTile, 113
- __getTileOptionsSubstring
 - HexTile, 113
- __getTileResourceSubstring
 - HexTile, 115
- __getTileTypeSubstring
 - HexTile, 115
- __getValidMapIndexPositions
 - HexMap, 89
- __handleImprovementStateMessage
 - Game, 67
- __handleKeyPressEvents
 - ContextMenu, 24
 - DieselGenerator, 42
 - EnergyStorageSystem, 54
 - Game, 68
 - HexMap, 90
 - HexTile, 116
 - Settlement, 160
 - SolarPV, 175
 - TidalTurbine, 192
 - TileImprovement, 208
 - WaveEnergyConverter, 233
 - WindTurbine, 251
- __handleKeyReleaseEvents
 - HexTile, 120
- __handleMouseButtonEvents
 - ContextMenu, 25
 - DieselGenerator, 43
 - EnergyStorageSystem, 55
 - Game, 68
 - HexMap, 90
 - HexTile, 121
 - Settlement, 161
 - SolarPV, 176
 - TidalTurbine, 193
 - TileImprovement, 208
 - WaveEnergyConverter, 234
 - WindTurbine, 252
- __insufficientCreditsAlarm
 - Game, 69
- __isClicked
 - HexTile, 121
- __isLakeTouchingOcean
 - HexMap, 91
- __layTiles
 - HexMap, 91
- __loadSoundBuffer
 - AssetsManager, 9
- __openBuildMenu
 - HexTile, 122
- __openProductionMenu
 - TileImprovement, 209
- __openUpgradeMenu
 - TileImprovement, 209
- __procedurallyGenerateTileResources
 - HexMap, 93
- __procedurallyGenerateTileTypes
 - HexMap, 94
- __processEvent
 - Game, 70
- __processMessage
 - Game, 70
- __repair
 - TileImprovement, 210
- __scrapImprovement
 - HexTile, 122
- __sendAssessNeighboursMessage
 - HexTile, 123
- __sendCreditsEarnedMessage
 - Game, 72
- __sendCreditsSpentMessage
 - HexTile, 123
 - TileImprovement, 210
- __sendGameStateMessage
 - Game, 72
- __sendGameStateRequest
 - HexTile, 124
 - TileImprovement, 211
- __sendImprovementStateMessage
 - DieselGenerator, 44
 - SolarPV, 176
 - TidalTurbine, 193
 - WaveEnergyConverter, 234
 - WindTurbine, 252
- __sendInsufficientCreditsMessage
 - HexTile, 124
 - TileImprovement, 211
- __sendNoTileSelectedMessage
 - HexMap, 94
- __sendQuitGameMessage
 - ContextMenu, 25
- __sendRestartGameMessage
 - ContextMenu, 25
- __sendTileSelectedMessage
 - HexTile, 124
- __sendTileStateMessage
 - HexTile, 125
- __sendTileStateRequest
 - TileImprovement, 211
- __sendTurnAdvanceMessage
 - Game, 73
- __sendUpdateGamePhaseMessage
 - HexTile, 125
- __setConsoleState
 - ContextMenu, 26
- __setConsoleString
 - ContextMenu, 26
- __setIsSelected
 - HexTile, 126
- __setResourceText
 - HexTile, 126
- __setUpBuildMenu
 - HexTile, 127

- __setUpBuildOption
 - HexTile, [128](#)
- __setUpCoinSprite
 - Settlement, [161](#)
- __setUpConsoleScreen
 - ContextMenu, [27](#)
- __setUpConsoleScreenFrame
 - ContextMenu, [27](#)
- __setUpDieselGeneratorBuildOption
 - HexTile, [129](#)
- __setUpDispatchIllustration
 - TileImprovement, [211](#)
- __setUpEnergyStorageSystemBuildOption
 - HexTile, [130](#)
- __setUpGlassScreen
 - HexMap, [95](#)
- __setUpMagnifyingGlassSprite
 - HexTile, [130](#)
- __setUpMenuFrame
 - ContextMenu, [29](#)
- __setUpNodeSprite
 - HexTile, [130](#)
- __setUpProductionMenu
 - EnergyStorageSystem, [55](#)
 - TileImprovement, [212](#)
- __setUpResourceChipSprite
 - HexTile, [131](#)
- __setUpSelectOutlineSprite
 - HexTile, [131](#)
- __setUpSolarPVBuildOption
 - HexTile, [131](#)
- __setUpTidalTurbineBuildOption
 - HexTile, [132](#)
- __setUpTileExplosionReel
 - HexTile, [132](#)
- __setUpTileImprovementSpriteAnimated
 - DieselGenerator, [44](#)
 - TidalTurbine, [193](#)
 - WaveEnergyConverter, [235](#)
 - WindTurbine, [253](#)
- __setUpTileImprovementSpriteStatic
 - EnergyStorageSystem, [56](#)
 - Settlement, [162](#)
 - SolarPV, [177](#)
- __setUpTileSprite
 - HexTile, [133](#)
- __setUpUpgradeMenu
 - TileImprovement, [212](#)
- __setUpVisualScreen
 - ContextMenu, [30](#)
- __setUpVisualScreenFrame
 - ContextMenu, [30](#)
- __setUpWaveEnergyConverterBuildOption
 - HexTile, [133](#)
- __setUpWindTurbineBuildOption
 - HexTile, [134](#)
- __smoothTileTypes
 - HexMap, [95](#)
- __toggleFrameClockOverlay
 - Game, [73](#)
- __upgrade
 - DieselGenerator, [44](#)
 - EnergyStorageSystem, [56](#)
- __upgradePowerCapacity
 - SolarPV, [177](#)
 - TidalTurbine, [194](#)
 - WaveEnergyConverter, [235](#)
 - WindTurbine, [253](#)
- __upgradeStorageCapacity
 - TileImprovement, [213](#)
- ~AssetsManager
 - AssetsManager, [8](#)
- ~ContextMenu
 - ContextMenu, [22](#)
- ~DieselGenerator
 - DieselGenerator, [41](#)
- ~EnergyStorageSystem
 - EnergyStorageSystem, [54](#)
- ~Game
 - Game, [63](#)
- ~HexMap
 - HexMap, [83](#)
- ~HexTile
 - HexTile, [107](#)
- ~MessageHub
 - MessageHub, [151](#)
- ~Settlement
 - Settlement, [160](#)
- ~SolarPV
 - SolarPV, [170](#)
- ~TidalTurbine
 - TidalTurbine, [187](#)
- ~TileImprovement
 - TileImprovement, [206](#)
- ~WaveEnergyConverter
 - WaveEnergyConverter, [228](#)
- ~WindTurbine
 - WindTurbine, [246](#)
- ABOVE_AVERAGE
 - HexTile.h, [294](#)
- addChannel
 - MessageHub, [151](#)
- advanceTurn
 - DieselGenerator, [45](#)
 - SolarPV, [178](#)
 - TidalTurbine, [194](#)
 - TileImprovement, [214](#)
 - WaveEnergyConverter, [236](#)
 - WindTurbine, [254](#)
- assess
 - HexMap, [95](#)
 - HexTile, [134](#)
- assets_manager_ptr
 - ContextMenu, [33](#)
 - Game, [75](#)
 - HexMap, [99](#)

- HexTile, 141
- TileImprovement, 218
- 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, 294
- BELOW_AVERAGE
 - HexTile.h, 294
- bobbing_y
 - TidalTurbine, 199
 - WaveEnergyConverter, 241
- bool_payload
 - Message, 148
- border_tiles_vec
 - HexMap, 99
- build_menu_backing
 - HexTile, 141
- build_menu_backing_text
 - HexTile, 142
- build_menu_open
 - HexTile, 142
- build_menu_options_text_vec
 - HexTile, 142
- build_menu_options_vec
 - HexTile, 142
- BUILD_SETTLEMENT
 - Game.h, 291
- BUILD_SETTLEMENT_COST
 - constants.h, 271
- capacity_factor_vec
 - SolarPV, 182
 - TidalTurbine, 199
 - WaveEnergyConverter, 241
- WindTurbine, 258
- capacity_kW
 - DieselGenerator, 50
 - SolarPV, 182
 - TidalTurbine, 199
 - WaveEnergyConverter, 241
 - WindTurbine, 258
- capacity_MWh
 - EnergyStorageSystem, 59
- channel
 - Message, 148
- charge_MWh
 - EnergyStorageSystem, 59
- check_terminating_conditions
 - Game, 75
- clear
 - AssetsManager, 10
 - HexMap, 96
 - MessageHub, 152
- CLEAR_FOREST_COST
 - constants.h, 271
- CLEAR_MOUNTAINS_COST
 - constants.h, 271
- CLEAR_PLAINS_COST
 - constants.h, 272
- clearMessages
 - MessageHub, 152
- clock
 - Game, 75
- coin_sprite
 - Settlement, 165
- 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, 262
- constants.h
 - BUILD_SETTLEMENT_COST, 271
 - CLEAR_FOREST_COST, 271
 - CLEAR_MOUNTAINS_COST, 271
 - CLEAR_PLAINS_COST, 272
 - COST_PER_LITRE_DIESEL, 272
 - CREDITS_PER_MWH_SERVED, 272

- DAILY_TIDAL_CAPACITY_FACTOR, [272](#)
- DIESEL_GENERATOR_BUILD_COST, [272](#)
- DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION, [272](#)
- EMISSIONS_LIFETIME_LIMIT_TONNES, [273](#)
- ENERGY_STORAGE_SYSTEM_BUILD_COST, [273](#)
- FLOAT_TOLERANCE, [273](#)
- FOREST_GREEN, [269](#)
- FRAMES_PER_SECOND, [273](#)
- GAME_CHANNEL, [273](#)
- GAME_HEIGHT, [273](#)
- GAME_STATE_CHANNEL, [274](#)
- GAME_WIDTH, [274](#)
- HEX_MAP_CHANNEL, [274](#)
- KG_CO2E_PER_LITRE_DIESEL, [274](#)
- LAKE_BLUE, [269](#)
- LITRES_DIESEL_PER_MWH_PRODUCTION, [274](#)
- MAX_STORAGE_LEVELS, [274](#)
- MAX_UPGRADE_LEVELS, [275](#)
- MAXIMUM_DAILY_DEMAND_PER_CAPITA, [275](#)
- MEAN_DAILY_DEMAND_RATIOS, [275](#)
- MEAN_DAILY_SOLAR_CAPACITY_FACTORS, [275](#)
- MEAN_DAILY_WAVE_CAPACITY_FACTORS, [275](#)
- MEAN_DAILY_WIND_CAPACITY_FACTORS, [276](#)
- MENU_FRAME_GREY, [269](#)
- MONOCHROME_SCREEN_BACKGROUND, [269](#)
- MONOCHROME_TEXT_AMBER, [269](#)
- MONOCHROME_TEXT_GREEN, [270](#)
- MONOCHROME_TEXT_RED, [270](#)
- MOUNTAINS_GREY, [270](#)
- NO_TILE_SELECTED_CHANNEL, [276](#)
- OCEAN_BLUE, [270](#)
- PLAINS_YELLOW, [270](#)
- POPULATION_MONTHLY_GROWTH_RATE, [276](#)
- RESOURCE_ASSESSMENT_COST, [276](#)
- RESOURCE_CHIP_GREY, [271](#)
- SCRAP_COST, [276](#)
- SECONDS_PER_FRAME, [277](#)
- SECONDS_PER_MONTH, [277](#)
- SECONDS_PER_YEAR, [277](#)
- SETTLEMENT_CHANNEL, [277](#)
- SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION, [277](#)
- SOLAR_PV_BUILD_COST, [277](#)
- SOLAR_PV_WATER_BUILD_MULTIPLIER, [278](#)
- STARTING_CREDITS, [278](#)
- STARTING_POPULATION, [278](#)
- STDEV_DAILY_DEMAND_RATIOS, [278](#)
- STDEV_DAILY_SOLAR_CAPACITY_FACTORS, [278](#)
- STDEV_DAILY_WAVE_CAPACITY_FACTORS, [279](#)
- STDEV_DAILY_WIND_CAPACITY_FACTORS, [279](#)
- TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION, [279](#)
- TIDAL_TURBINE_BUILD_COST, [279](#)
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES, [280](#)
- TILE_SELECTED_CHANNEL, [280](#)
- TILE_STATE_CHANNEL, [280](#)
- TILE_TYPE_CUMULATIVE_PROBABILITIES, [280](#)
- VISUAL_SCREEN_FRAME_GREY, [271](#)
- WAVE_ENERGY_CONVERTER_BUILD_COST, [280](#)
- WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION, [281](#)
- WIND_OP_MAINT_COST_PER_MWH_PRODUCTION, [281](#)
- WIND_TURBINE_BUILD_COST, [281](#)
- WIND_TURBINE_WATER_BUILD_MULTIPLIER, [281](#)
- constructRenderWindow
 - main.cpp, [311](#)
- context_menu_ptr
 - Game, [75](#)
- 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, [262](#)
 - MENU, [262](#)
 - N_CONSOLE_STATES, [262](#)
 - NONE_STATE, [262](#)
 - READY, [262](#)
 - TILE, [262](#)
- COST_PER_LITRE_DIESEL
 - constants.h, [272](#)
- credits
 - Game, [76](#)
 - HexTile, [142](#)
 - TileImprovement, [218](#)
- CREDITS_PER_MWH_SERVED
 - constants.h, [272](#)
- cumulative_emissions_tonnes
 - Game, [76](#)
- current_track
 - AssetsManager, [18](#)
- DAILY_TIDAL_CAPACITY_FACTOR
 - constants.h, [272](#)
- decorateTile
 - HexTile, [135](#)
- decoration_cleared
 - HexTile, [142](#)
- demand_MWh
 - Game, [76](#)
 - TileImprovement, [218](#)
- demand_remaining_MWh
 - Game, [76](#)
- demand_vec_MWh
 - Game, [76](#)
 - TileImprovement, [218](#)
- DIESEL_GENERATOR
 - TileImprovement.h, [299](#)
- DIESEL_GENERATOR_BUILD_COST
 - constants.h, [272](#)
- DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION
 - constants.h, [272](#)
- DieselGenerator, [37](#)
 - __breakdown, [41](#)
 - __computeProductionCosts, [41](#)
 - __drawProductionMenu, [41](#)
 - __handleKeyPressEvents, [42](#)
 - __handleMouseButtonEvents, [43](#)
 - __sendImprovementStateMessage, [44](#)
 - __setUpTileImprovementSpriteAnimated, [44](#)
 - __upgrade, [44](#)
 - ~DieselGenerator, [41](#)
 - advanceTurn, [45](#)
 - capacity_kW, [50](#)
 - DieselGenerator, [39](#)
 - draw, [46](#)
 - emissions_tonnes_CO2e, [50](#)
 - fuel_cost, [50](#)
 - getTileOptionsSubstring, [48](#)
 - max_production_MWh, [50](#)
 - processEvent, [49](#)
 - processMessage, [49](#)
 - production_MWh, [50](#)
 - setIsSelected, [49](#)
 - smoke_da, [51](#)
 - smoke_dx, [51](#)
 - smoke_dy, [51](#)
 - smoke_prob, [51](#)
 - smoke_sprite_list, [51](#)
- dispatch_backing
 - TileImprovement, [218](#)
- dispatch_MWh
 - SolarPV, [182](#)
 - TidalTurbine, [199](#)
 - WaveEnergyConverter, [241](#)
 - WindTurbine, [259](#)
- dispatch_text
 - TileImprovement, [218](#)
- dispatch_vec_MWh
 - SolarPV, [182](#)
 - TidalTurbine, [200](#)
 - WaveEnergyConverter, [241](#)
 - WindTurbine, [259](#)
- dispatchable_MWh
 - SolarPV, [182](#)
 - TidalTurbine, [200](#)
 - WaveEnergyConverter, [241](#)
 - WindTurbine, [259](#)
- double_payload
 - Message, [148](#)
- draw
 - ContextMenu, [31](#)
 - DieselGenerator, [46](#)
 - EnergyStorageSystem, [57](#)
 - HexMap, [96](#)
 - HexTile, [136](#)
 - Settlement, [162](#)
 - SolarPV, [178](#)
 - TidalTurbine, [195](#)
 - TileImprovement, [214](#)
 - WaveEnergyConverter, [237](#)
 - WindTurbine, [255](#)
- draw_coin
 - Settlement, [165](#)
- draw_explosion
 - HexTile, [143](#)
- EMISSIONS_LIFETIME_LIMIT_TONNES
 - constants.h, [273](#)
- emissions_tonnes_CO2e
 - DieselGenerator, [50](#)
- ENERGY_STORAGE_SYSTEM
 - TileImprovement.h, [299](#)

ENERGY_STORAGE_SYSTEM_BUILD_COST
 constants.h, 273

EnergyStorageSystem, 52
 __handleKeyPressEvents, 54
 __handleMouseButtonEvents, 55
 __setUpProductionMenu, 55
 __setUpTileImprovementSpriteStatic, 56
 __upgrade, 56
 ~EnergyStorageSystem, 54
 capacity_MWh, 59
 charge_MWh, 59
 draw, 57
 EnergyStorageSystem, 53
 getTileOptionsSubstring, 57
 processEvent, 58
 processMessage, 58
 setIsSelected, 59

event
 Game, 76

event_ptr
 ContextMenu, 35
 HexMap, 99
 HexTile, 143
 TileImprovement, 219

expectedErrorNotDetected
 testing_utils.cpp, 304
 testing_utils.h, 284

explosion_frame
 HexTile, 143

explosion_sprite_reel
 HexTile, 143

FLOAT_TOLERANCE
 constants.h, 273

font_map
 AssetsManager, 18

FOREST
 HexTile.h, 294

FOREST_GREEN
 constants.h, 269

frame
 ContextMenu, 35
 Game, 77
 HexMap, 99
 HexTile, 143
 TileImprovement, 219

FRAMES_PER_SECOND
 constants.h, 273

fuel_cost
 DieselGenerator, 50

Game, 60
 __advanceTurn, 64
 __checkTerminatingConditions, 64
 __computeCurrentDemand, 64
 __draw, 65
 __drawFrameClockOverlay, 65
 __drawHUD, 66
 __handleImprovementStateMessage, 67
 __handleKeyPressEvents, 68
 __handleMouseButtonEvents, 68
 __insufficientCreditsAlarm, 69
 __processEvent, 70
 __processMessage, 70
 __sendCreditsEarnedMessage, 72
 __sendGameStateMessage, 72
 __sendTurnAdvanceMessage, 73
 __toggleFrameClockOverlay, 73
 ~Game, 63
 assets_manager_ptr, 75
 check_terminating_conditions, 75
 clock, 75
 context_menu_ptr, 75
 credits, 76
 cumulative_emissions_tonnes, 76
 demand_MWh, 76
 demand_remaining_MWh, 76
 demand_vec_MWh, 76
 event, 76
 frame, 77
 Game, 63
 game_loop_broken, 77
 game_phase, 77
 hex_map_ptr, 77
 message_deadlock, 77
 message_deadlock_frame, 77
 message_hub, 78
 month, 78
 population, 78
 quit_game, 78
 render_window_ptr, 78
 run, 74
 show_frame_clock_overlay, 78
 time_since_start_s, 79
 turn, 79
 year, 79

Game.h
 BUILD_SETTLEMENT, 291
 GamePhase, 291
 LOSS_CREDITS, 291
 LOSS_DEMAND, 291
 LOSS_EMISSIONS, 291
 N_GAME_PHASES, 291
 SYSTEM_MANAGEMENT, 291
 VICTORY, 291

GAME_CHANNEL
 constants.h, 273

GAME_HEIGHT
 constants.h, 273

game_loop_broken
 Game, 77

game_menu_up
 ContextMenu, 35

game_phase
 Game, 77
 HexTile, 143
 TileImprovement, 219

GAME_STATE_CHANNEL
 constants.h, 274
 GAME_WIDTH
 constants.h, 274
 GamePhase
 Game.h, 291
 getCurrentTrackKey
 AssetsManager, 11
 getFont
 AssetsManager, 11
 getSound
 AssetsManager, 12
 getSoundBuffer
 AssetsManager, 12
 getTexture
 AssetsManager, 13
 getTileOptionsSubstring
 DieselGenerator, 48
 EnergyStorageSystem, 57
 Settlement, 163
 SolarPV, 179
 TidalTurbine, 197
 TileImprovement, 216
 WaveEnergyConverter, 238
 WindTurbine, 256
 getTrackStatus
 AssetsManager, 13
 glass_screen
 HexMap, 99
 GOOD
 HexTile.h, 294

 has_improvement
 HexTile, 144
 hasTraffic
 MessageHub, 152
 header/ContextMenu.h, 261
 header/DieselGenerator.h, 262
 header/EnergyStorageSystem.h, 263
 header/ESC_core/AssetsManager.h, 264
 header/ESC_core/constants.h, 265
 header/ESC_core/doxygen_cite.h, 281
 header/ESC_core/includes.h, 282
 header/ESC_core/MessageHub.h, 283
 header/ESC_core/testing_utils.h, 283
 header/Game.h, 290
 header/HexMap.h, 291
 header/HexTile.h, 292
 header/Settlement.h, 294
 header/SolarPV.h, 295
 header/TidalTurbine.h, 296
 header/TileImprovement.h, 297
 header/WaveEnergyConverter.h, 299
 header/WindTurbine.h, 300
 health
 TileImprovement, 219
 hex_draw_order_vec
 HexMap, 100
 hex_map
 HexMap, 100
 HEX_MAP_CHANNEL
 constants.h, 274
 hex_map_ptr
 Game, 77
 HexMap, 79
 __assembleHexMap, 83
 __assessNeighbours, 83
 __buildDrawOrderVector, 84
 __enforceOceanContinuity, 84
 __getMajorityTileType, 85
 __getNeighboursVector, 86
 __getNoise, 87
 __getSelectedTile, 88
 __getValidMapIndexPositions, 89
 __handleKeyPressEvents, 90
 __handleMouseButtonEvents, 90
 __isLakeTouchingOcean, 91
 __layTiles, 91
 __procedurallyGenerateTileResources, 93
 __procedurallyGenerateTileTypes, 94
 __sendNoTileSelectedMessage, 94
 __setUpGlassScreen, 95
 __smoothTileTypes, 95
 ~HexMap, 83
 assess, 95
 assets_manager_ptr, 99
 border_tiles_vec, 99
 clear, 96
 draw, 96
 event_ptr, 99
 frame, 99
 glass_screen, 99
 hex_draw_order_vec, 100
 hex_map, 100
 HexMap, 82
 message_hub_ptr, 100
 n_layers, 100
 n_tiles, 100
 position_x, 100
 position_y, 101
 processEvent, 97
 processMessage, 97
 render_window_ptr, 101
 reroll, 98
 show_resource, 101
 tile_position_x_vec, 101
 tile_position_y_vec, 101
 tile_selected, 101
 toggleResourceOverlay, 98
 HexTile, 102
 __buildDieselGenerator, 108
 __buildEnergyStorage, 108
 __buildSettlement, 109
 __buildSolarPV, 109
 __buildTidalTurbine, 110
 __buildWaveEnergyConverter, 110
 __buildWindTurbine, 111

- __clearDecoration, 112
- __closeBuildMenu, 112
- __getTileCoordsSubstring, 112
- __getTileImprovementSubstring, 113
- __getTileOptionsSubstring, 113
- __getTileResourceSubstring, 115
- __getTileTypeSubstring, 115
- __handleKeyPressEvents, 116
- __handleKeyReleaseEvents, 120
- __handleMouseButtonEvents, 121
- __isClicked, 121
- __openBuildMenu, 122
- __scrapImprovement, 122
- __sendAssessNeighboursMessage, 123
- __sendCreditsSpentMessage, 123
- __sendGameStateRequest, 124
- __sendInsufficientCreditsMessage, 124
- __sendTileSelectedMessage, 124
- __sendTileStateMessage, 125
- __sendUpdateGamePhaseMessage, 125
- __setIsSelected, 126
- __setResourceText, 126
- __setUpBuildMenu, 127
- __setUpBuildOption, 128
- __setUpDieselGeneratorBuildOption, 129
- __setUpEnergyStorageSystemBuildOption, 130
- __setUpMagnifyingGlassSprite, 130
- __setUpNodeSprite, 130
- __setUpResourceChipSprite, 131
- __setUpSelectOutlineSprite, 131
- __setUpSolarPVBuildOption, 131
- __setUpTidalTurbineBuildOption, 132
- __setUpTileExplosionReel, 132
- __setUpTileSprite, 133
- __setUpWaveEnergyConverterBuildOption, 133
- __setUpWindTurbineBuildOption, 134
- ~HexTile, 107
- assess, 134
- assets_manager_ptr, 141
- build_menu_backing, 141
- build_menu_backing_text, 142
- build_menu_open, 142
- build_menu_options_text_vec, 142
- build_menu_options_vec, 142
- credits, 142
- decorateTile, 135
- decoration_cleared, 142
- draw, 136
- draw_explosion, 143
- event_ptr, 143
- explosion_frame, 143
- explosion_sprite_reel, 143
- frame, 143
- game_phase, 143
- has_improvement, 144
- HexTile, 106
- is_selected, 144
- magnifying_glass_sprite, 144
- major_radius, 144
- message_hub_ptr, 144
- minor_radius, 144
- node_sprite, 145
- position_x, 145
- position_y, 145
- processEvent, 137
- processMessage, 138
- render_window_ptr, 145
- resource_assessed, 145
- resource_assessment, 145
- resource_chip_sprite, 146
- resource_text, 146
- scrap_improvement_frame, 146
- select_outline_sprite, 146
- setTileResource, 138, 139
- setTileType, 139, 140
- show_node, 146
- show_resource, 146
- tile_decoration_sprite, 147
- tile_improvement_ptr, 147
- tile_resource, 147
- tile_sprite, 147
- tile_type, 147
- toggleResourceOverlay, 141
- HexTile.h
 - ABOVE_AVERAGE, 294
 - AVERAGE, 294
 - BELOW_AVERAGE, 294
 - FOREST, 294
 - GOOD, 294
 - LAKE, 294
 - MOUNTAINS, 294
 - N_TILE_RESOURCES, 294
 - N_TILE_TYPES, 294
 - NONE_TYPE, 294
 - OCEAN, 294
 - PLAINS, 294
 - POOR, 294
 - TileResource, 293
 - TileType, 294
- int_payload
 - Message, 149
- is_broken
 - TileImprovement, 219
- is_running
 - TileImprovement, 219
- is_selected
 - HexTile, 144
 - TileImprovement, 220
- isEmpty
 - MessageHub, 152
- just_built
 - TileImprovement, 220
- just_upgraded
 - TileImprovement, 220

- KG_CO2E_PER_LITRE_DIESEL
 - constants.h, [274](#)
- LAKE
 - HexTile.h, [294](#)
- LAKE_BLUE
 - constants.h, [269](#)
- LITRES_DIESEL_PER_MWH_PRODUCTION
 - constants.h, [274](#)
- loadAssets
 - main.cpp, [311](#)
- loadFont
 - AssetsManager, [14](#)
- loadSound
 - AssetsManager, [14](#)
- loadTexture
 - AssetsManager, [15](#)
- loadTrack
 - AssetsManager, [16](#)
- LOSS_CREDITS
 - Game.h, [291](#)
- LOSS_DEMAND
 - Game.h, [291](#)
- LOSS_EMISSIONS
 - Game.h, [291](#)
- magnifying_glass_sprite
 - HexTile, [144](#)
- main
 - main.cpp, [314](#)
- main.cpp
 - constructRenderWindow, [311](#)
 - loadAssets, [311](#)
 - main, [314](#)
- major_radius
 - HexTile, [144](#)
- max_daily_production_MWh
 - SolarPV, [183](#)
 - TidalTurbine, [200](#)
 - WaveEnergyConverter, [242](#)
 - WindTurbine, [259](#)
- max_production_MWh
 - DieselGenerator, [50](#)
- MAX_STORAGE_LEVELS
 - constants.h, [274](#)
- MAX_UPGRADE_LEVELS
 - constants.h, [275](#)
- MAXIMUM_DAILY_DEMAND_PER_CAPITA
 - constants.h, [275](#)
- MEAN_DAILY_DEMAND_RATIOS
 - constants.h, [275](#)
- MEAN_DAILY_SOLAR_CAPACITY_FACTORS
 - constants.h, [275](#)
- MEAN_DAILY_WAVE_CAPACITY_FACTORS
 - constants.h, [275](#)
- MEAN_DAILY_WIND_CAPACITY_FACTORS
 - constants.h, [276](#)
- MENU
 - ContextMenu.h, [262](#)
- menu_frame
 - ContextMenu, [35](#)
- MENU_FRAME_GREY
 - constants.h, [269](#)
- Message, [148](#)
 - bool_payload, [148](#)
 - channel, [148](#)
 - double_payload, [148](#)
 - int_payload, [149](#)
 - string_payload, [149](#)
 - subject, [149](#)
 - vector_payload, [149](#)
- message_deadlock
 - Game, [77](#)
- message_deadlock_frame
 - Game, [77](#)
- message_hub
 - Game, [78](#)
- message_hub_ptr
 - ContextMenu, [35](#)
 - HexMap, [100](#)
 - HexTile, [144](#)
 - TileImprovement, [220](#)
- message_map
 - MessageHub, [157](#)
- MessageHub, [149](#)
 - ~MessageHub, [151](#)
 - addChannel, [151](#)
 - clear, [152](#)
 - clearMessages, [152](#)
 - hasTraffic, [152](#)
 - isEmpty, [152](#)
 - message_map, [157](#)
 - MessageHub, [150](#)
 - popMessage, [154](#)
 - printState, [155](#)
 - receiveMessage, [155](#)
 - removeChannel, [156](#)
 - sendMessage, [156](#)
- minor_radius
 - HexTile, [144](#)
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, [269](#)
- MONOCHROME_TEXT_AMBER
 - constants.h, [269](#)
- MONOCHROME_TEXT_GREEN
 - constants.h, [270](#)
- MONOCHROME_TEXT_RED
 - constants.h, [270](#)
- month
 - Game, [78](#)
 - TileImprovement, [220](#)
- MOUNTAINS
 - HexTile.h, [294](#)
- MOUNTAINS_GREY
 - constants.h, [270](#)
- N_CONSOLE_STATES
 - ContextMenu.h, [262](#)

- N_GAME_PHASES
 - Game.h, [291](#)
- n_layers
 - HexMap, [100](#)
- N_TILE_IMPROVEMENT_TYPES
 - TileImprovement.h, [299](#)
- N_TILE_RESOURCES
 - HexTile.h, [294](#)
- N_TILE_TYPES
 - HexTile.h, [294](#)
- n_tiles
 - HexMap, [100](#)
- nextTrack
 - AssetsManager, [16](#)
- NO_TILE_SELECTED_CHANNEL
 - constants.h, [276](#)
- node_sprite
 - HexTile, [145](#)
- NONE_STATE
 - ContextMenu.h, [262](#)
- NONE_TYPE
 - HexTile.h, [294](#)
- OCEAN
 - HexTile.h, [294](#)
- OCEAN_BLUE
 - constants.h, [270](#)
- operation_maintenance_cost
 - TileImprovement, [220](#)
- pauseTrack
 - AssetsManager, [17](#)
- PLAINS
 - HexTile.h, [294](#)
- PLAINS_YELLOW
 - constants.h, [270](#)
- playTrack
 - AssetsManager, [17](#)
- POOR
 - HexTile.h, [294](#)
- popMessage
 - MessageHub, [154](#)
- population
 - Game, [78](#)
- POPULATION_MONTHLY_GROWTH_RATE
 - constants.h, [276](#)
- position_x
 - ContextMenu, [36](#)
 - HexMap, [100](#)
 - HexTile, [145](#)
 - TileImprovement, [221](#)
- position_y
 - ContextMenu, [36](#)
 - HexMap, [101](#)
 - HexTile, [145](#)
 - TileImprovement, [221](#)
- previousTrack
 - AssetsManager, [17](#)
- printGold
 - testing_utils.cpp, [304](#)
 - testing_utils.h, [285](#)
- printGreen
 - testing_utils.cpp, [304](#)
 - testing_utils.h, [285](#)
- printRed
 - testing_utils.cpp, [305](#)
 - testing_utils.h, [286](#)
- printState
 - MessageHub, [155](#)
- processEvent
 - ContextMenu, [32](#)
 - DieselGenerator, [49](#)
 - EnergyStorageSystem, [58](#)
 - HexMap, [97](#)
 - HexTile, [137](#)
 - Settlement, [164](#)
 - SolarPV, [180](#)
 - TidalTurbine, [197](#)
 - TileImprovement, [216](#)
 - WaveEnergyConverter, [239](#)
 - WindTurbine, [257](#)
- processMessage
 - ContextMenu, [32](#)
 - DieselGenerator, [49](#)
 - EnergyStorageSystem, [58](#)
 - HexMap, [97](#)
 - HexTile, [138](#)
 - Settlement, [164](#)
 - SolarPV, [181](#)
 - TidalTurbine, [198](#)
 - TileImprovement, [216](#)
 - WaveEnergyConverter, [239](#)
 - WindTurbine, [257](#)
- production_menu_backing
 - TileImprovement, [221](#)
- production_menu_backing_text
 - TileImprovement, [221](#)
- production_menu_open
 - TileImprovement, [221](#)
- production_MWh
 - DieselGenerator, [50](#)
 - SolarPV, [183](#)
 - TidalTurbine, [200](#)
 - WaveEnergyConverter, [242](#)
 - WindTurbine, [259](#)
- production_vec_MWh
 - SolarPV, [183](#)
 - TidalTurbine, [200](#)
 - WaveEnergyConverter, [242](#)
 - WindTurbine, [259](#)
- quit_game
 - Game, [78](#)
- READY
 - ContextMenu.h, [262](#)
- receiveMessage
 - MessageHub, [155](#)

- removeChannel
 - MessageHub, 156
- render_window_ptr
 - ContextMenu, 36
 - Game, 78
 - HexMap, 101
 - HexTile, 145
 - TileImprovement, 221
- reroll
 - HexMap, 98
- resource_assessed
 - HexTile, 145
- resource_assessment
 - HexTile, 145
- RESOURCE_ASSESSMENT_COST
 - constants.h, 276
- RESOURCE_CHIP_GREY
 - constants.h, 271
- resource_chip_sprite
 - HexTile, 146
- resource_text
 - HexTile, 146
- rotor_drotation
 - TidalTurbine, 200
 - WindTurbine, 260
- run
 - Game, 74
- SCRAP_COST
 - constants.h, 276
- scrap_improvement_frame
 - HexTile, 146
- SECONDS_PER_FRAME
 - constants.h, 277
- SECONDS_PER_MONTH
 - constants.h, 277
- SECONDS_PER_YEAR
 - constants.h, 277
- select_outline_sprite
 - HexTile, 146
- sendMessage
 - MessageHub, 156
- setIsSelected
 - DieselGenerator, 49
 - EnergyStorageSystem, 59
 - Settlement, 164
 - SolarPV, 181
 - TidalTurbine, 198
 - TileImprovement, 217
 - WaveEnergyConverter, 240
 - WindTurbine, 257
- setTileResource
 - HexTile, 138, 139
- setTileType
 - HexTile, 139, 140
- SETTLEMENT
 - TileImprovement.h, 299
- Settlement, 157
 - __handleKeyPressEvents, 160
 - __handleMouseButtonEvents, 161
 - __setUpCoinSprite, 161
 - __setUpTileImprovementSpriteStatic, 162
 - ~Settlement, 160
 - coin_sprite, 165
 - draw, 162
 - draw_coin, 165
 - getTileOptionsSubstring, 163
 - processEvent, 164
 - processMessage, 164
 - setIsSelected, 164
 - Settlement, 159
 - smoke_da, 165
 - smoke_dx, 165
 - smoke_dy, 166
 - smoke_prob, 166
 - smoke_sprite_list, 166
- SETTLEMENT_CHANNEL
 - constants.h, 277
- show_frame_clock_overlay
 - Game, 78
- show_node
 - HexTile, 146
- show_resource
 - HexMap, 101
 - HexTile, 146
- smoke_da
 - DieselGenerator, 51
 - Settlement, 165
- smoke_dx
 - DieselGenerator, 51
 - Settlement, 165
- smoke_dy
 - DieselGenerator, 51
 - Settlement, 166
- smoke_prob
 - DieselGenerator, 51
 - Settlement, 166
- smoke_sprite_list
 - DieselGenerator, 51
 - Settlement, 166
- SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION
 - constants.h, 277
- SOLAR_PV
 - TileImprovement.h, 299
- SOLAR_PV_BUILD_COST
 - constants.h, 277
- SOLAR_PV_WATER_BUILD_MULTIPLIER
 - constants.h, 278
- SolarPV, 167
 - __breakdown, 170
 - __computeCapacityFactors, 171
 - __computeDispatch, 171
 - __computeProduction, 172
 - __computeProductionCosts, 172
 - __drawProductionMenu, 173
 - __drawUpgradeOptions, 173
 - __handleKeyPressEvents, 175

- __handleMouseButtonEvents, 176
- __sendImprovementStateMessage, 176
- __setUpTileImprovementSpriteStatic, 177
- __upgradePowerCapacity, 177
- ~SolarPV, 170
- advanceTurn, 178
- capacity_factor_vec, 182
- capacity_kW, 182
- dispatch_MWh, 182
- dispatch_vec_MWh, 182
- dispatchable_MWh, 182
- draw, 178
- getTileOptionsSubstring, 179
- max_daily_production_MWh, 183
- processEvent, 180
- processMessage, 181
- production_MWh, 183
- production_vec_MWh, 183
- setIsSelected, 181
- SolarPV, 169
- update, 181
- sound_map
 - AssetsManager, 18
- soundbuffer_map
 - AssetsManager, 18
- source/ContextMenu.cpp, 301
- source/DieselGenerator.cpp, 301
- source/EnergyStorageSystem.cpp, 302
- source/ESC_core/AssetsManager.cpp, 302
- source/ESC_core/MessageHub.cpp, 302
- source/ESC_core/testing_utils.cpp, 303
- source/Game.cpp, 309
- source/HexMap.cpp, 309
- source/HexTile.cpp, 310
- source/main.cpp, 310
- source/Settlement.cpp, 315
- source/SolarPV.cpp, 315
- source/TidalTurbine.cpp, 316
- source/TileImprovement.cpp, 316
- source/WaveEnergyConverter.cpp, 316
- source/WindTurbine.cpp, 317
- STARTING_CREDITS
 - constants.h, 278
- STARTING_POPULATION
 - constants.h, 278
- STDEV_DAILY_DEMAND_RATIOS
 - constants.h, 278
- STDEV_DAILY_SOLAR_CAPACITY_FACTORS
 - constants.h, 278
- STDEV_DAILY_WAVE_CAPACITY_FACTORS
 - constants.h, 279
- STDEV_DAILY_WIND_CAPACITY_FACTORS
 - constants.h, 279
- stopTrack
 - AssetsManager, 17
- storage_kWh
 - TileImprovement, 222
- storage_level
 - TileImprovement, 222
- storage_upgrade_sprite
 - TileImprovement, 222
- storage_upgrade_sprite_vec
 - TileImprovement, 222
- string_payload
 - Message, 149
- subject
 - Message, 149
- SYSTEM_MANAGEMENT
 - Game.h, 291
- testFloatEquals
 - testing_utils.cpp, 305
 - testing_utils.h, 286
- testGreaterThan
 - testing_utils.cpp, 306
 - testing_utils.h, 287
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 306
 - testing_utils.h, 287
- testing_utils.cpp
 - expectedErrorNotDetected, 304
 - printGold, 304
 - printGreen, 304
 - printRed, 305
 - testFloatEquals, 305
 - testGreaterThan, 306
 - testGreaterThanOrEqualTo, 306
 - testLessThan, 307
 - testLessThanOrEqualTo, 308
 - testTruth, 308
- testing_utils.h
 - expectedErrorNotDetected, 284
 - printGold, 285
 - printGreen, 285
 - printRed, 286
 - testFloatEquals, 286
 - testGreaterThan, 287
 - testGreaterThanOrEqualTo, 287
 - testLessThan, 288
 - testLessThanOrEqualTo, 289
 - testTruth, 289
- testLessThan
 - testing_utils.cpp, 307
 - testing_utils.h, 288
- testLessThanOrEqualTo
 - testing_utils.cpp, 308
 - testing_utils.h, 289
- testTruth
 - testing_utils.cpp, 308
 - testing_utils.h, 289
- texture_map
 - AssetsManager, 18
- TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION
 - constants.h, 279
- TIDAL_TURBINE
 - TileImprovement.h, 299
- TIDAL_TURBINE_BUILD_COST

- constants.h, 279
- TidalTurbine, 184
 - __breakdown, 187
 - __computeCapacityFactors, 188
 - __computeDispatch, 188
 - __computeProduction, 189
 - __computeProductionCosts, 189
 - __drawProductionMenu, 189
 - __drawUpgradeOptions, 190
 - __handleKeyPressEvents, 192
 - __handleMouseButtonEvents, 193
 - __sendImprovementStateMessage, 193
 - __setUpTileImprovementSpriteAnimated, 193
 - __upgradePowerCapacity, 194
 - ~TidalTurbine, 187
- advanceTurn, 194
- bobbing_y, 199
- capacity_factor_vec, 199
- capacity_kW, 199
- dispatch_MWh, 199
- dispatch_vec_MWh, 200
- dispatchable_MWh, 200
- draw, 195
- getTileOptionsSubstring, 197
- max_daily_production_MWh, 200
- processEvent, 197
- processMessage, 198
- production_MWh, 200
- production_vec_MWh, 200
- rotor_drotation, 200
- setIsSelected, 198
- TidalTurbine, 186
- update, 199
- TILE
 - ContextMenu.h, 262
- tile_decoration_sprite
 - HexTile, 147
- tile_improvement_ptr
 - HexTile, 147
- tile_improvement_sprite_animated
 - TileImprovement, 222
- tile_improvement_sprite_static
 - TileImprovement, 222
- tile_improvement_string
 - TileImprovement, 223
- tile_improvement_type
 - TileImprovement, 223
- tile_position_x_vec
 - HexMap, 101
- tile_position_y_vec
 - HexMap, 101
- tile_resource
 - HexTile, 147
 - TileImprovement, 223
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES
 - constants.h, 280
- tile_resource_scalar
 - TileImprovement, 223
- tile_selected
 - HexMap, 101
- TILE_SELECTED_CHANNEL
 - constants.h, 280
- tile_sprite
 - HexTile, 147
- TILE_STATE_CHANNEL
 - constants.h, 280
- tile_type
 - HexTile, 147
- TILE_TYPE_CUMULATIVE_PROBABILITIES
 - constants.h, 280
- TileImprovement, 201
 - __breakdown, 207
 - __closeProductionMenu, 207
 - __closeUpgradeMenu, 207
 - __drawDispatch, 207
 - __handleKeyPressEvents, 208
 - __handleMouseButtonEvents, 208
 - __openProductionMenu, 209
 - __openUpgradeMenu, 209
 - __repair, 210
 - __sendCreditsSpentMessage, 210
 - __sendGameStateRequest, 211
 - __sendInsufficientCreditsMessage, 211
 - __sendTileStateRequest, 211
 - __setUpDispatchIllustration, 211
 - __setUpProductionMenu, 212
 - __setUpUpgradeMenu, 212
 - __upgradeStorageCapacity, 213
 - ~TileImprovement, 206
- advanceTurn, 214
- assets_manager_ptr, 218
- credits, 218
- demand_MWh, 218
- demand_vec_MWh, 218
- dispatch_backing, 218
- dispatch_text, 218
- draw, 214
- event_ptr, 219
- frame, 219
- game_phase, 219
- getTileOptionsSubstring, 216
- health, 219
- is_broken, 219
- is_running, 219
- is_selected, 220
- just_built, 220
- just_upgraded, 220
- message_hub_ptr, 220
- month, 220
- operation_maintenance_cost, 220
- position_x, 221
- position_y, 221
- processEvent, 216
- processMessage, 216
- production_menu_backing, 221
- production_menu_backing_text, 221

- production_menu_open, [221](#)
- render_window_ptr, [221](#)
- setIsSelected, [217](#)
- storage_kWh, [222](#)
- storage_level, [222](#)
- storage_upgrade_sprite, [222](#)
- storage_upgrade_sprite_vec, [222](#)
- tile_improvement_sprite_animated, [222](#)
- tile_improvement_sprite_static, [222](#)
- tile_improvement_string, [223](#)
- tile_improvement_type, [223](#)
- tile_resource, [223](#)
- tile_resource_scalar, [223](#)
- TileImprovement, [205](#)
- update, [217](#)
- upgrade_arrow_sprite, [223](#)
- upgrade_frame, [223](#)
- upgrade_level, [224](#)
- upgrade_menu_backing, [224](#)
- upgrade_menu_backing_text, [224](#)
- upgrade_menu_open, [224](#)
- upgrade_plus_sprite, [224](#)
- TileImprovement.h
 - DIESEL_GENERATOR, [299](#)
 - ENERGY_STORAGE_SYSTEM, [299](#)
 - N_TILE_IMPROVEMENT_TYPES, [299](#)
 - SETTLEMENT, [299](#)
 - SOLAR_PV, [299](#)
 - TIDAL_TURBINE, [299](#)
 - TileImprovementType, [298](#)
 - WAVE_ENERGY_CONVERTER, [299](#)
 - WIND_TURBINE, [299](#)
- TileImprovementType
 - TileImprovement.h, [298](#)
- TileResource
 - HexTile.h, [293](#)
- TileType
 - HexTile.h, [294](#)
- time_since_start_s
 - Game, [79](#)
- toggleResourceOverlay
 - HexMap, [98](#)
 - HexTile, [141](#)
- track_map
 - AssetsManager, [19](#)
- turn
 - Game, [79](#)
- update
 - SolarPV, [181](#)
 - TidalTurbine, [199](#)
 - TileImprovement, [217](#)
 - WaveEnergyConverter, [240](#)
 - WindTurbine, [258](#)
- upgrade_arrow_sprite
 - TileImprovement, [223](#)
- upgrade_frame
 - TileImprovement, [223](#)
- upgrade_level
 - TileImprovement, [224](#)
- upgrade_menu_backing
 - TileImprovement, [224](#)
- upgrade_menu_backing_text
 - TileImprovement, [224](#)
- upgrade_menu_open
 - TileImprovement, [224](#)
- upgrade_plus_sprite
 - TileImprovement, [224](#)
- vector_payload
 - Message, [149](#)
- VICTORY
 - Game.h, [291](#)
- visual_screen
 - ContextMenu, [36](#)
- visual_screen_frame_bottom
 - ContextMenu, [36](#)
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, [271](#)
- visual_screen_frame_left
 - ContextMenu, [36](#)
- visual_screen_frame_right
 - ContextMenu, [37](#)
- visual_screen_frame_top
 - ContextMenu, [37](#)
- WAVE_ENERGY_CONVERTER
 - TileImprovement.h, [299](#)
- WAVE_ENERGY_CONVERTER_BUILD_COST
 - constants.h, [280](#)
- WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION
 - constants.h, [281](#)
- WaveEnergyConverter, [225](#)
 - __breakdown, [228](#)
 - __computeCapacityFactors, [229](#)
 - __computeDispatch, [229](#)
 - __computeProduction, [230](#)
 - __computeProductionCosts, [231](#)
 - __drawProductionMenu, [231](#)
 - __drawUpgradeOptions, [232](#)
 - __handleKeyPressEvents, [233](#)
 - __handleMouseButtonEvents, [234](#)
 - __sendImprovementStateMessage, [234](#)
 - __setUpTileImprovementSpriteAnimated, [235](#)
 - __upgradePowerCapacity, [235](#)
 - ~WaveEnergyConverter, [228](#)
 - advanceTurn, [236](#)
 - bobbing_y, [241](#)
 - capacity_factor_vec, [241](#)
 - capacity_kW, [241](#)
 - dispatch_MWh, [241](#)
 - dispatch_vec_MWh, [241](#)
 - dispatchable_MWh, [241](#)
 - draw, [237](#)
 - getTileOptionsSubstring, [238](#)
 - max_daily_production_MWh, [242](#)
 - processEvent, [239](#)
 - processMessage, [239](#)

- production_MWh, [242](#)
- production_vec_MWh, [242](#)
- setIsSelected, [240](#)
- update, [240](#)
- WaveEnergyConverter, [227](#)
- WIND_OP_MAINT_COST_PER_MWH_PRODUCTION
 - constants.h, [281](#)
- WIND_TURBINE
 - TileImprovement.h, [299](#)
- WIND_TURBINE_BUILD_COST
 - constants.h, [281](#)
- WIND_TURBINE_WATER_BUILD_MULTIPLIER
 - constants.h, [281](#)
- WindTurbine, [243](#)
 - __breakdown, [246](#)
 - __computeCapacityFactors, [247](#)
 - __computeDispatch, [247](#)
 - __computeProduction, [248](#)
 - __computeProductionCosts, [249](#)
 - __drawProductionMenu, [249](#)
 - __drawUpgradeOptions, [250](#)
 - __handleKeyPressEvents, [251](#)
 - __handleMouseButtonEvents, [252](#)
 - __sendImprovementStateMessage, [252](#)
 - __setUpTileImprovementSpriteAnimated, [253](#)
 - __upgradePowerCapacity, [253](#)
 - ~WindTurbine, [246](#)
 - advanceTurn, [254](#)
 - capacity_factor_vec, [258](#)
 - capacity_kW, [258](#)
 - dispatch_MWh, [259](#)
 - dispatch_vec_MWh, [259](#)
 - dispatchable_MWh, [259](#)
 - draw, [255](#)
 - getTileOptionsSubstring, [256](#)
 - max_daily_production_MWh, [259](#)
 - processEvent, [257](#)
 - processMessage, [257](#)
 - production_MWh, [259](#)
 - production_vec_MWh, [259](#)
 - rotor_drotation, [260](#)
 - setIsSelected, [257](#)
 - update, [258](#)
 - WindTurbine, [245](#)
- year
 - Game, [79](#)