

Road To Zero

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 AssetsManager Class Reference	5
3.1.1 Detailed Description	6
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 AssetsManager()	6
3.1.2.2 ~AssetsManager()	7
3.1.3 Member Function Documentation	7
3.1.3.1 __loadSoundBuffer()	7
3.1.3.2 clear()	8
3.1.3.3 getCurrentTrackKey()	9
3.1.3.4 getFont()	9
3.1.3.5 getSound()	10
3.1.3.6 getSoundBuffer()	10
3.1.3.7 getTexture()	11
3.1.3.8 getTrackStatus()	11
3.1.3.9 loadFont()	12
3.1.3.10 loadSound()	12
3.1.3.11 loadTexture()	13
3.1.3.12 loadTrack()	14
3.1.3.13 nextTrack()	15
3.1.3.14 pauseTrack()	15
3.1.3.15 playTrack()	15
3.1.3.16 previousTrack()	15
3.1.3.17 stopTrack()	16
3.1.4 Member Data Documentation	16
3.1.4.1 current_track	16
3.1.4.2 font_map	16
3.1.4.3 sound_map	16
3.1.4.4 soundbuffer_map	16
3.1.4.5 texture_map	17
3.1.4.6 track_map	17
3.2 ContextMenu Class Reference	17
3.2.1 Detailed Description	19
3.2.2 Constructor & Destructor Documentation	19
3.2.2.1 ContextMenu()	19
3.2.2.2 ~ContextMenu()	20
3.2.3 Member Function Documentation	20

3.2.3.1	__drawConsoleScreenFrame()	20
3.2.3.2	__drawConsoleText()	21
3.2.3.3	__drawVisualScreenFrame()	22
3.2.3.4	__handleKeyPressEvents()	22
3.2.3.5	__handleMouseButtonEvents()	22
3.2.3.6	__sendQuitGameMessage()	23
3.2.3.7	__sendRestartGameMessage()	23
3.2.3.8	__setConsoleState()	23
3.2.3.9	__setConsoleString()	24
3.2.3.10	__setUpConsoleScreen()	25
3.2.3.11	__setUpConsoleScreenFrame()	25
3.2.3.12	__setUpMenuFrame()	27
3.2.3.13	__setUpVisualScreen()	27
3.2.3.14	__setUpVisualScreenFrame()	28
3.2.3.15	draw()	29
3.2.3.16	processEvent()	29
3.2.3.17	processMessage()	30
3.2.4	Member Data Documentation	31
3.2.4.1	assets_manager_ptr	31
3.2.4.2	console_screen	31
3.2.4.3	console_screen_frame_bottom	31
3.2.4.4	console_screen_frame_left	31
3.2.4.5	console_screen_frame_right	31
3.2.4.6	console_screen_frame_top	32
3.2.4.7	console_state	32
3.2.4.8	console_string	32
3.2.4.9	event_ptr	32
3.2.4.10	frame	32
3.2.4.11	game_menu_up	32
3.2.4.12	menu_frame	33
3.2.4.13	message_hub_ptr	33
3.2.4.14	position_x	33
3.2.4.15	position_y	33
3.2.4.16	render_window_ptr	33
3.2.4.17	visual_screen	33
3.2.4.18	visual_screen_frame_bottom	34
3.2.4.19	visual_screen_frame_left	34
3.2.4.20	visual_screen_frame_right	34
3.2.4.21	visual_screen_frame_top	34
3.3	Game Class Reference	34
3.3.1	Detailed Description	36
3.3.2	Constructor & Destructor Documentation	36

3.3.2.1 Game()	37
3.3.2.2 ~Game()	37
3.3.3 Member Function Documentation	38
3.3.3.1 __draw()	38
3.3.3.2 __drawFrameClockOverlay()	38
3.3.3.3 __drawHUD()	39
3.3.3.4 __handleKeyPressEvents()	39
3.3.3.5 __handleMouseButtonEvents()	40
3.3.3.6 __insufficientCreditsAlarm()	40
3.3.3.7 __processEvent()	41
3.3.3.8 __processMessage()	42
3.3.3.9 __sendGameStateMessage()	43
3.3.3.10 __toggleFrameClockOverlay()	43
3.3.3.11 run()	43
3.3.4 Member Data Documentation	44
3.3.4.1 assets_manager_ptr	44
3.3.4.2 clock	44
3.3.4.3 context_menu_ptr	45
3.3.4.4 credits	45
3.3.4.5 cumulative_emissions_tonnes	45
3.3.4.6 demand_MWh	45
3.3.4.7 event	45
3.3.4.8 frame	45
3.3.4.9 game_loop_broken	46
3.3.4.10 hex_map_ptr	46
3.3.4.11 message_hub	46
3.3.4.12 month	46
3.3.4.13 population	46
3.3.4.14 quit_game	46
3.3.4.15 render_window_ptr	47
3.3.4.16 show_frame_clock_overlay	47
3.3.4.17 time_since_start_s	47
3.3.4.18 year	47
3.4 HexMap Class Reference	47
3.4.1 Detailed Description	50
3.4.2 Constructor & Destructor Documentation	50
3.4.2.1 HexMap()	50
3.4.2.2 ~HexMap()	51
3.4.3 Member Function Documentation	51
3.4.3.1 __assembleHexMap()	51
3.4.3.2 __buildDrawOrderVector()	51
3.4.3.3 __enforceOceanContinuity()	52

3.4.3.4	__getMajorityTileType()	53
3.4.3.5	__getNeighboursVector()	54
3.4.3.6	__getNoise()	54
3.4.3.7	__getSelectedTile()	56
3.4.3.8	__getValidMapIndexPositions()	56
3.4.3.9	__handleKeyPressEvents()	57
3.4.3.10	__handleMouseButtonEvents()	58
3.4.3.11	__isLakeTouchingOcean()	58
3.4.3.12	__layTiles()	59
3.4.3.13	__procedurallyGenerateTileResources()	61
3.4.3.14	__procedurallyGenerateTileTypes()	61
3.4.3.15	__sendNoTileSelectedMessage()	62
3.4.3.16	__setUpGlassScreen()	62
3.4.3.17	__smoothTileTypes()	63
3.4.3.18	assess()	63
3.4.3.19	clear()	63
3.4.3.20	draw()	64
3.4.3.21	processEvent()	64
3.4.3.22	processMessage()	65
3.4.3.23	reroll()	65
3.4.3.24	toggleResourceOverlay()	66
3.4.4	Member Data Documentation	66
3.4.4.1	assets_manager_ptr	66
3.4.4.2	border_tiles_vec	66
3.4.4.3	event_ptr	66
3.4.4.4	frame	67
3.4.4.5	glass_screen	67
3.4.4.6	hex_draw_order_vec	67
3.4.4.7	hex_map	67
3.4.4.8	message_hub_ptr	67
3.4.4.9	n_layers	67
3.4.4.10	n_tiles	68
3.4.4.11	position_x	68
3.4.4.12	position_y	68
3.4.4.13	render_window_ptr	68
3.4.4.14	tile_position_x_vec	68
3.4.4.15	tile_position_y_vec	68
3.4.4.16	tile_selected	69
3.5	HexTile Class Reference	69
3.5.1	Detailed Description	72
3.5.2	Constructor & Destructor Documentation	72
3.5.2.1	HexTile()	72

3.5.2.2 ~HexTile()	73
3.5.3 Member Function Documentation	73
3.5.3.1 __getTileCoordsSubstring()	73
3.5.3.2 __getTileImprovementSubstring()	74
3.5.3.3 __getTileOptionsSubstring()	74
3.5.3.4 __getTileResourceSubstring()	75
3.5.3.5 __getTileTypeSubstring()	75
3.5.3.6 __handleKeyPressEvents()	76
3.5.3.7 __handleMouseButtonEvents()	77
3.5.3.8 __isClicked()	77
3.5.3.9 __sendCreditsSpentMessage()	78
3.5.3.10 __sendGameStateRequest()	78
3.5.3.11 __sendInsufficientCreditsMessage()	78
3.5.3.12 __sendTileSelectedMessage()	79
3.5.3.13 __sendTileStateMessage()	79
3.5.3.14 __setResourceText()	79
3.5.3.15 __setUpNodeSprite()	80
3.5.3.16 __setUpResourceChipSprite()	81
3.5.3.17 __setUpSelectOutlineSprite()	81
3.5.3.18 __setUpTileSprite()	82
3.5.3.19 assess()	82
3.5.3.20 decorateTile()	82
3.5.3.21 draw()	83
3.5.3.22 processEvent()	84
3.5.3.23 processMessage()	84
3.5.3.24 setTileResource() [1/2]	85
3.5.3.25 setTileResource() [2/2]	86
3.5.3.26 setTileType() [1/2]	86
3.5.3.27 setTileType() [2/2]	87
3.5.3.28 toggleResourceOverlay()	87
3.5.4 Member Data Documentation	88
3.5.4.1 assets_manager_ptr	88
3.5.4.2 credits	88
3.5.4.3 event_ptr	88
3.5.4.4 frame	88
3.5.4.5 has_improvement	88
3.5.4.6 is_selected	89
3.5.4.7 major_radius	89
3.5.4.8 message_hub_ptr	89
3.5.4.9 minor_radius	89
3.5.4.10 node_sprite	89
3.5.4.11 position_x	89

3.5.4.12 position_y	90
3.5.4.13 render_window_ptr	90
3.5.4.14 resource_assessed	90
3.5.4.15 resource_chip_sprite	90
3.5.4.16 resource_text	90
3.5.4.17 select_outline_sprite	90
3.5.4.18 settlement_built	91
3.5.4.19 show_node	91
3.5.4.20 show_resource	91
3.5.4.21 tile_decoration_sprite	91
3.5.4.22 tile_improvement_ptr	91
3.5.4.23 tile_resource	91
3.5.4.24 tile_sprite	92
3.5.4.25 tile_type	92
3.6 Message Struct Reference	92
3.6.1 Detailed Description	92
3.6.2 Member Data Documentation	92
3.6.2.1 bool_payload	93
3.6.2.2 channel	93
3.6.2.3 double_payload	93
3.6.2.4 int_payload	93
3.6.2.5 string_payload	93
3.6.2.6 subject	93
3.7 MessageHub Class Reference	94
3.7.1 Detailed Description	94
3.7.2 Constructor & Destructor Documentation	94
3.7.2.1 MessageHub()	95
3.7.2.2 ~MessageHub()	95
3.7.3 Member Function Documentation	95
3.7.3.1 addChannel()	95
3.7.3.2 clear()	96
3.7.3.3 clearMessages()	96
3.7.3.4 hasTraffic()	96
3.7.3.5 isEmpty()	97
3.7.3.6 popMessage()	98
3.7.3.7 receiveMessage()	99
3.7.3.8 removeChannel()	99
3.7.3.9 sendMessage()	100
3.7.4 Member Data Documentation	100
3.7.4.1 message_map	101
3.8 TileImprovement Class Reference	101
3.8.1 Detailed Description	102

3.8.2 Constructor & Destructor Documentation	102
3.8.2.1 TileImprovement()	102
3.8.2.2 ~TileImprovement()	103
3.8.3 Member Function Documentation	103
3.8.3.1 __handleKeyPressEvents()	103
3.8.3.2 __handleMouseButtonEvents()	104
3.8.3.3 draw()	104
3.8.3.4 processEvent()	104
3.8.3.5 processMessage()	105
3.8.4 Member Data Documentation	105
3.8.4.1 assets_manager_ptr	105
3.8.4.2 credits	105
3.8.4.3 event_ptr	105
3.8.4.4 frame	105
3.8.4.5 message_hub_ptr	106
3.8.4.6 position_x	106
3.8.4.7 position_y	106
3.8.4.8 render_window_ptr	106
3.8.4.9 tile_improvement_sprite_animated	106
3.8.4.10 tile_improvement_sprite_static	106
4 File Documentation	107
4.1 header/ContextMenu.h File Reference	107
4.1.1 Detailed Description	108
4.1.2 Enumeration Type Documentation	108
4.1.2.1 ConsoleState	108
4.2 header/ESC_core/AssetsManager.h File Reference	108
4.2.1 Detailed Description	109
4.3 header/ESC_core/constants.h File Reference	109
4.3.1 Detailed Description	111
4.3.2 Function Documentation	111
4.3.2.1 FOREST_GREEN()	111
4.3.2.2 LAKE_BLUE()	112
4.3.2.3 MENU_FRAME_GREY()	112
4.3.2.4 MONOCHROME_SCREEN_BACKGROUND()	112
4.3.2.5 MONOCHROME_TEXT_AMBER()	112
4.3.2.6 MONOCHROME_TEXT_GREEN()	112
4.3.2.7 MONOCHROME_TEXT_RED()	113
4.3.2.8 MOUNTAINS_GREY()	113
4.3.2.9 OCEAN_BLUE()	113
4.3.2.10 PLAINS_YELLOW()	113
4.3.2.11 RESOURCE_CHIP_GREY()	113

4.3.2.12 VISUAL_SCREEN_FRAME_GREY()	114
4.3.3 Variable Documentation	114
4.3.3.1 CO2E_KG_PER_LITRE_DIESEL	114
4.3.3.2 EMISSIONS_LIFETIME_LIMIT_TONNES	114
4.3.3.3 FLOAT_TOLERANCE	114
4.3.3.4 FRAMES_PER_SECOND	114
4.3.3.5 GAME_CHANNEL	115
4.3.3.6 GAME_HEIGHT	115
4.3.3.7 GAME_STATE_CHANNEL	115
4.3.3.8 GAME_WIDTH	115
4.3.3.9 NO_TILE_SELECTED_CHANNEL	115
4.3.3.10 RESOURCE_ASSESSMENT_COST	115
4.3.3.11 SECONDS_PER_FRAME	116
4.3.3.12 SECONDS_PER_MONTH	116
4.3.3.13 SECONDS_PER_YEAR	116
4.3.3.14 TILE_RESOURCE_CUMULATIVE_PROBABILITIES	116
4.3.3.15 TILE_SELECTED_CHANNEL	116
4.3.3.16 TILE_STATE_CHANNEL	116
4.3.3.17 TILE_TYPE_CUMULATIVE_PROBABILITIES	117
4.4 header/ESC_core/doxygen_cite.h File Reference	117
4.4.1 Detailed Description	117
4.5 header/ESC_core/includes.h File Reference	117
4.5.1 Detailed Description	118
4.6 header/ESC_core/MessageHub.h File Reference	118
4.6.1 Detailed Description	119
4.7 header/ESC_core/testing_utils.h File Reference	119
4.7.1 Detailed Description	120
4.7.2 Function Documentation	120
4.7.2.1 expectedErrorNotDetected()	121
4.7.2.2 printGold()	121
4.7.2.3 printGreen()	121
4.7.2.4 printRed()	122
4.7.2.5 testFloatEquals()	122
4.7.2.6 testGreaterThan()	123
4.7.2.7 testGreaterThanOrEqualTo()	123
4.7.2.8 testLessThan()	124
4.7.2.9 testLessThanOrEqualTo()	125
4.7.2.10 testTruth()	125
4.8 header/Game.h File Reference	126
4.9 header/HexMap.h File Reference	127
4.9.1 Detailed Description	127
4.10 header/HexTile.h File Reference	128

4.10.1 Detailed Description	129
4.10.2 Enumeration Type Documentation	129
4.10.2.1 TileResource	129
4.10.2.2 TileType	129
4.11 header/TileImprovement.h File Reference	130
4.11.1 Detailed Description	131
4.11.2 Enumeration Type Documentation	131
4.11.2.1 TileImprovementType	131
4.12 source/ContextMenu.cpp File Reference	131
4.12.1 Detailed Description	132
4.13 source/ESC_core/AssetsManager.cpp File Reference	132
4.13.1 Detailed Description	132
4.14 source/ESC_core/MessageHub.cpp File Reference	132
4.14.1 Detailed Description	133
4.15 source/ESC_core/testing_utils.cpp File Reference	133
4.15.1 Detailed Description	133
4.15.2 Function Documentation	134
4.15.2.1 expectedErrorNotDetected()	134
4.15.2.2 printGold()	134
4.15.2.3 printGreen()	134
4.15.2.4 printRed()	135
4.15.2.5 testFloatEquals()	135
4.15.2.6 testGreaterThan()	136
4.15.2.7 testGreaterThanOrEqualTo()	136
4.15.2.8 testLessThan()	137
4.15.2.9 testLessThanOrEqualTo()	138
4.15.2.10 testTruth()	138
4.16 source/Game.cpp File Reference	139
4.16.1 Detailed Description	139
4.17 source/HexMap.cpp File Reference	139
4.17.1 Detailed Description	140
4.18 source/HexTile.cpp File Reference	140
4.18.1 Detailed Description	140
4.19 source/main.cpp File Reference	140
4.19.1 Detailed Description	141
4.19.2 Function Documentation	141
4.19.2.1 constructRenderWindow()	141
4.19.2.2 loadAssets()	141
4.19.2.3 main()	142
4.20 source/TileImprovement.cpp File Reference	142
4.20.1 Detailed Description	142

Bibliography	143
Index	145

Chapter 1

Class Index

1.1 Class List

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

AssetsManager	A class which manages visual and sound assets	5
ContextMenu	A class which defines a context menu for the game	17
Game	A class which acts as the central class for the game, by containing all other classes and implementing the game loop	34
HexMap	A class which defines a hex map of hex tiles	47
HexTile	A class which defines a hex tile of the hex map	69
Message	A structure which defines a standard message format	92
MessageHub	A class which acts as a central hub for inter-object message traffic	94
TileImprovement	A base class for the tile improvement hierarchy	101

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

header/ ContextMenu.h	
Header file for the ContextMenu class	107
header/ Game.h	126
header/ HexMap.h	
Header file for the HexMap class	127
header/ HexTile.h	
Header file for the Game class	128
header/ TileImprovement.h	
Header file for the TileImprovement class	130
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	108
header/ESC_core/ constants.h	
Header file for various constants	109
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	117
header/ESC_core/ includes.h	
Header file for various includes	117
header/ESC_core/ MessageHub.h	
Header file for the MessageHub class	118
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	119
source/ ContextMenu.cpp	
Implementation file for the ContextMenu class	131
source/ Game.cpp	
Implementation file for the Game class	139
source/ HexMap.cpp	
Implementation file for the HexMap class	139
source/ HexTile.cpp	
Implementation file for the HexTile class	140
source/ main.cpp	
Implementation file for main() for Road To Zero	140
source/ TileImprovement.cpp	
Implementation file for the TileImprovement class	142
source/ESC_core/ AssetsManager.cpp	
Implementation file for the AssetsManager class	132

source/ESC_core/ MessageHub.cpp	
Implementation file for the MessageHub class	132
source/ESC_core/ testing_utils.cpp	
Implementation file for various testing utilities	133

Chapter 3

Class Documentation

3.1 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.

3.1.1 Detailed Description

A class which manages visual and sound assets.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AssetsManager()

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

Constructor for the [AssetsManager](#) class.

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

3.1.2.2 ~AssetsManager()

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

Destructor for the [AssetsManager](#) class.

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

3.1.3 Member Function Documentation

3.1.3.1 __loadSoundBuffer()

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

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

Parameters

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

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

```

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

```

3.1.3.2 clear()

```

void AssetsManager::clear (
    void )

```

Method to clear all loaded assets.

```

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

```

```

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

```

3.1.3.3 getCurrentTrackKey()

```

std::string AssetsManager::getCurrentTrackKey (
    void )

```

Method to get track key for current track.

Returns

The track key for the current track.

```

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

```

3.1.3.4 getFont()

```

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

```

Method to get font associated with given font key.

Parameters

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

Returns

A pointer to the corresponding font.

```

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

```

```

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

```

3.1.3.5 getSound()

```

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

```

Method to get sound associated with given sound key.

Parameters

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

Returns

A pointer to the corresponding sound.

```

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

```

3.1.3.6 getSoundBuffer()

```

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

```

Method to get soundbuffer associated with given sound key.

Parameters

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

Returns

A pointer to the corresponding soundbuffer.

```

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

```

3.1.3.7 getTexture()

```

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

```

Method to get texture associated with given texture key.

Parameters

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

Returns

A pointer to the corresponding texture.

```

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

```

3.1.3.8 getTrackStatus()

```

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

```

Method to get the status of the current track.

Returns

The status of the current track.

```

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

```

3.1.3.9 loadFont()

```

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

```

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

Parameters

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

```

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

```

3.1.3.10 loadSound()

```

void AssetsManager::loadSound (

```



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

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

Parameters

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

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

3.1.3.11 loadTexture()

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

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

Parameters

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

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

```

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

```

3.1.3.12 loadTrack()

```

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

```

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

Parameters

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

```

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

```

3.1.3.13 nextTrack()

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

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

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

3.1.3.14 pauseTrack()

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

Method to pause the current track.

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

3.1.3.15 playTrack()

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

Method to play the current track.

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

3.1.3.16 previousTrack()

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

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

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

3.1.3.17 stopTrack()

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

Method to stop the current track.

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

3.1.4 Member Data Documentation

3.1.4.1 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).

3.1.4.2 font_map

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

A map of pointers to loaded fonts.

3.1.4.3 sound_map

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

A map of pointers to loaded sounds.

3.1.4.4 soundbuffer_map

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

A map of pointers to sound buffers.

3.1.4.5 texture_map

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

A map of pointers to loaded textures.

3.1.4.6 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:

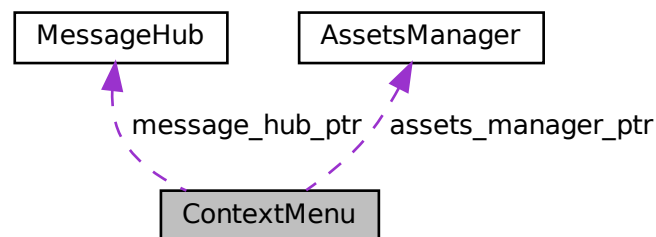
- header/ESC_core/[AssetsManager.h](#)
- source/ESC_core/[AssetsManager.cpp](#)

3.2 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 [game_menu_up](#)
Indicates whether or not the game menu is up.
- 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.

3.2.1 Detailed Description

A class which defines a context menu for the game.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 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.

```

784 {
785     // 1. set attributes
786
787     // 1.1. private
788     this->event_ptr = event_ptr;
789     this->render_window_ptr = render_window_ptr;
790
791     this->assets_manager_ptr = assets_manager_ptr;
792     this->message_hub_ptr = message_hub_ptr;
793
794     // 1.2. public
795     this->console_state = ConsoleState :: NONE_STATE;
796     this->__setConsoleState(ConsoleState :: READY);
797
798     this->game_menu_up = false;
799
800     this->frame = 0;
801
802     this->position_x = GAME_WIDTH;
803     this->position_y = 0;
804
805     // 2. set up and position drawable attributes
806     this->__setUpMenuFrame();
807     this->__setUpVisualScreen();
808     this->__setUpVisualScreenFrame();
809     this->__setUpConsoleScreen();
810     this->__setUpConsoleScreenFrame();
811
812     std::cout << "ContextMenu constructed at " << this << std::endl;
813
814     return;
815 } /* ContextMenu() */

```

3.2.2.2 ~ContextMenu()

```

ContextMenu::~ContextMenu (
    void )

```

Destructor for the [ContextMenu](#) class.

```

962 {
963     std::cout << "ContextMenu at " << this << " destroyed" << std::endl;
964
965     return;
966 } /* ~ContextMenu() */

```

3.2.3 Member Function Documentation

3.2.3.1 __drawConsoleScreenFrame()

```

void ContextMenu::__drawConsoleScreenFrame (
    void ) [private]

```

Helper method to draw console screen frame.

```

433 {

```



```

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

```

3.2.3.2 __drawConsoleText()

```

void ContextMenu::__drawConsoleText (
    void ) [private]

```

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

```

550 {
551     // 1. set up console text (drawable)
552     sf::Text console_text(
553         this->console_string,
554         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
555         16
556     );
557
558     console_text.setFillColor(MONOCROME_TEXT_GREEN);
559
560     console_text.setPosition(
561         this->position_x - 50 - 300 + 16,
562         this->position_y + GAME_HEIGHT - 50 - 340 + 16
563     );
564
565
566     // 2. draw console text
567     this->render_window_ptr->draw(console_text);
568
569
570     // 3. assemble and draw blinking console cursor
571     if ((this->frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
572         sf::RectangleShape console_cursor(sf::Vector2f(10, 16));
573
574         console_cursor.setFillColor(MONOCROME_TEXT_GREEN);
575
576         console_cursor.setPosition(
577             console_text.getPosition().x,
578             console_text.getPosition().y + console_text.getLocalBounds().height + 10
579         );
580
581         this->render_window_ptr->draw(console_cursor);
582     }
583
584     // 4. updating frame count if console is in menu state
585     if (this->console_state == ConsoleState::MENU) {
586         std::string frame_count_string = "FRAME: ";
587         frame_count_string += std::to_string(this->frame);
588
589         sf::Text frame_count_text(
590             frame_count_string,
591             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
592             16
593         );
594
595         frame_count_text.setFillColor(MONOCROME_TEXT_GREEN);
596
597         frame_count_text.setPosition(
598             console_text.getPosition().x,
599             console_text.getPosition().y + console_text.getLocalBounds().height - 10
600         );
601
602         this->render_window_ptr->draw(frame_count_text);
603     }
604
605     return;
606 } /* __drawConsoleText() */

```

3.2.3.3 __drawVisualScreenFrame()

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

Helper method to draw visual screen frame.

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

3.2.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
621 {
622     switch (this->event_ptr->key.code) {
623         case (sf::Keyboard::Escape): {
624             if (this->console_state == ConsoleState :: MENU) {
625                 this->__setConsoleState(ConsoleState :: READY);
626             }
627
628             else {
629                 this->__setConsoleState(ConsoleState :: MENU);
630             }
631
632             break;
633         }
634
635         case (sf::Keyboard::Q): {
636             if (this->console_state == ConsoleState :: MENU) {
637                 this->__sendQuitGameMessage();
638             }
639
640             }
641
642         case (sf::Keyboard::R): {
643             if (this->console_state == ConsoleState :: MENU) {
644                 this->__sendRestartGameMessage();
645             }
646
647             }
648
649         default: {
650             // do nothing!
651
652             break;
653         }
654     }
655 }
656
657 return;
658 } /* __handleKeyPressEvents() */
```

3.2.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```

673 {
674     switch (this->event_ptr->mouseButton.button) {
675         case (sf::Mouse::Left): {
676             //...
677
678             break;
679         }
680
681         case (sf::Mouse::Right): {
682             //...
683
684             break;
685         }
686     }
687
688     default: {
689         // do nothing!
690
691         break;
692     }
693 }
694 }
695
696 return;
697 } /* __handleMouseButtonEvents() */

```

3.2.3.6 __sendQuitGameMessage()

```

void ContextMenu::__sendQuitGameMessage (
    void ) [private]

```

Helper method to format and send a quit game message.

```

712 {
713     Message quit_game_message;
714
715     quit_game_message.channel = GAME_CHANNEL;
716     quit_game_message.subject = "quit game";
717
718     this->message_hub_ptr->sendMessage(quit_game_message);
719
720     std::cout << "Quit game message sent by " << this << std::endl;
721     return;
722 } /* __sendQuitGameMessage() */

```

3.2.3.7 __sendRestartGameMessage()

```

void ContextMenu::__sendRestartGameMessage (
    void ) [private]

```

Helper method to format and send a restart game message.

```

737 {
738     Message restart_game_message;
739
740     restart_game_message.channel = GAME_CHANNEL;
741     restart_game_message.subject = "restart game";
742
743     this->message_hub_ptr->sendMessage(restart_game_message);
744
745     std::cout << "Restart game message sent by " << this << std::endl;
746     return;
747 } /* __sendRestartGameMessage() */

```

3.2.3.8 __setConsoleState()

```

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

```

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

Parameters

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

```

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

```

3.2.3.9 __setConsoleString()

```

void ContextMenu::__setConsoleString (
    void ) [private]

```

Helper method to set console string depending on console state.

```

483 {
484     this->console_string.clear();
485
486     switch (this->console_state) {
487         case (ConsoleState :: MENU): {
488             // 32 char x 17 line console "-----\n";
489             this->console_string = "      **** MENU **** \n";
490             this->console_string += " \n";
491             this->console_string += "[R]:  RESTART \n";
492             this->console_string += " \n";
493             this->console_string += "[TAB]: TOGGLE RESOURCE OVERLAY \n";
494             this->console_string += "[T]:  TUTORIAL \n";
495             this->console_string += " \n";
496             this->console_string += " \n";
497             this->console_string += " \n";
498             this->console_string += " \n";
499             this->console_string += " \n";
500             this->console_string += " \n";
501             this->console_string += " \n";
502             this->console_string += "[Q]:   QUIT \n";
503             this->console_string += "[ESC]: CLOSE MENU \n";
504             this->console_string += " \n";
505
506             break;
507         }
508
509         case (ConsoleState :: TILE): {
510             // take console string from tile state message
511
512             break;
513         }
514
515         default: {
516             // 32 char x 17 line console "-----\n";
517             this->console_string = "      **** RTZ 64 CONTEXT V12 **** \n";
518             this->console_string += " \n";
519             this->console_string += "64K RAM SYSTEM  38911 BYTES FREE \n";
520             this->console_string += " \n";
521             this->console_string += "[TAB]: TOGGLE RESOURCE OVERLAY \n";
522             this->console_string += " \n";
523             this->console_string += "[ESC]:          MENU \n";
524             this->console_string += "[LEFT CLICK]: TILE INFO/OPTIONS \n";
525             this->console_string += " \n";
526             this->console_string += " \n";
527             this->console_string += " \n";
528             this->console_string += "READY. \n";
529
530             break;
531         }
532     }
533
534     return;
535 } /* __setConsoleString() */

```

3.2.3.10 __setUpConsoleScreen()

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

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

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

3.2.3.11 __setUpConsoleScreenFrame()

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

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

```
255 {
256     int n_points = 4;
257
258     // 1. top framing
259     this->console_screen_frame_top.setPointCount(n_points);
260
261     this->console_screen_frame_top.setPoint(
262         0,
263         sf::Vector2f(
264             this->position_x - 50,
265             this->position_y + GAME_HEIGHT - 50 - 340
266         )
267     );
268     this->console_screen_frame_top.setPoint(
269         1,
270         sf::Vector2f(
271             this->position_x - 50 + 16,
272             this->position_y + GAME_HEIGHT - 50 - 340 - 16
273         )
274     );
275     this->console_screen_frame_top.setPoint(
276         2,
277         sf::Vector2f(
278             this->position_x - 350 - 16,
279             this->position_y + GAME_HEIGHT - 50 - 340 - 16
280         )
281     );
282     this->console_screen_frame_top.setPoint(
283         3,
284         sf::Vector2f(
285             this->position_x - 350,
286             this->position_y + GAME_HEIGHT - 50 - 340
287         )
288     );
289
290     this->console_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
291
292     this->console_screen_frame_top.setOutlineThickness(2);
293     this->console_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
294
295     this->console_screen_frame_top.move(0, -2);
296
297
298     // 2. left framing
299     this->console_screen_frame_left.setPointCount(n_points);
300
301     this->console_screen_frame_left.setPoint(
302         0,
303         sf::Vector2f(
304             this->position_x - 350,
305             this->position_y + GAME_HEIGHT - 50 - 340
```

```

306     )
307 );
308 this->console_screen_frame_left.setPoint(
309     1,
310     sf::Vector2f(
311         this->position_x - 350 - 16,
312         this->position_y + GAME_HEIGHT - 50 - 340 - 16
313     )
314 );
315 this->console_screen_frame_left.setPoint(
316     2,
317     sf::Vector2f(
318         this->position_x - 350 - 16,
319         this->position_y + GAME_HEIGHT - 50 + 16
320     )
321 );
322 this->console_screen_frame_left.setPoint(
323     3,
324     sf::Vector2f(
325         this->position_x - 350,
326         this->position_y + GAME_HEIGHT - 50
327     )
328 );
329
330 this->console_screen_frame_left.setFillColors(VISUAL_SCREEN_FRAME_GREY);
331
332 this->console_screen_frame_left.setOutlineThickness(2);
333 this->console_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
334
335 this->console_screen_frame_left.move(-2, 0);
336
337
338 // 3. bottom framing
339 this->console_screen_frame_bottom.setPointCount(n_points);
340
341 this->console_screen_frame_bottom.setPoint(
342     0,
343     sf::Vector2f(
344         this->position_x - 350,
345         this->position_y + GAME_HEIGHT - 50
346     )
347 );
348 this->console_screen_frame_bottom.setPoint(
349     1,
350     sf::Vector2f(
351         this->position_x - 350 - 16,
352         this->position_y + GAME_HEIGHT - 50 + 16
353     )
354 );
355 this->console_screen_frame_bottom.setPoint(
356     2,
357     sf::Vector2f(
358         this->position_x - 50 + 16,
359         this->position_y + GAME_HEIGHT - 50 + 16
360     )
361 );
362 this->console_screen_frame_bottom.setPoint(
363     3,
364     sf::Vector2f(
365         this->position_x - 50,
366         this->position_y + GAME_HEIGHT - 50
367     )
368 );
369
370 this->console_screen_frame_bottom.setFillColors(VISUAL_SCREEN_FRAME_GREY);
371
372 this->console_screen_frame_bottom.setOutlineThickness(2);
373 this->console_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
374
375 this->console_screen_frame_bottom.move(0, 2);
376
377
378 // 4. right framing
379 this->console_screen_frame_right.setPointCount(n_points);
380
381 this->console_screen_frame_right.setPoint(
382     0,
383     sf::Vector2f(
384         this->position_x - 50,
385         this->position_y + GAME_HEIGHT - 50
386     )
387 );
388 this->console_screen_frame_right.setPoint(
389     1,
390     sf::Vector2f(
391         this->position_x - 50 + 16,
392         this->position_y + GAME_HEIGHT - 50 + 16

```

```

393     )
394 );
395 this->console_screen_frame_right.setPoint(
396     2,
397     sf::Vector2f(
398         this->position_x - 50 + 16,
399         this->position_y + GAME_HEIGHT - 50 - 340 - 16
400     )
401 );
402 this->console_screen_frame_right.setPoint(
403     3,
404     sf::Vector2f(
405         this->position_x - 50,
406         this->position_y + GAME_HEIGHT - 50 - 340
407     )
408 );
409
410 this->console_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
411
412 this->console_screen_frame_right.setOutlineThickness(2);
413 this->console_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
414
415 this->console_screen_frame_right.move(2, 0);
416
417 return;
418 } /* __setUpConsoleScreenFrame() */

```

3.2.3.12 __setUpMenuFrame()

```

void ContextMenu::__setUpMenuFrame (
    void ) [private]

```

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

```

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

```

3.2.3.13 __setUpVisualScreen()

```

void ContextMenu::__setUpVisualScreen (
    void ) [private]

```

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

```

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

```

3.2.3.14 __setUpVisualScreenFrame()

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

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

```
78 {
79     int n_points = 4;
80
81     // 1. top framing
82     this->visual_screen_frame_top.setPointCount(n_points);
83
84     this->visual_screen_frame_top.setPoint(
85         0,
86         sf::Vector2f(this->position_x - 50, this->position_y + 50)
87     );
88     this->visual_screen_frame_top.setPoint(
89         1,
90         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
91     );
92     this->visual_screen_frame_top.setPoint(
93         2,
94         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
95     );
96     this->visual_screen_frame_top.setPoint(
97         3,
98         sf::Vector2f(this->position_x - 350, this->position_y + 50)
99     );
100
101     this->visual_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
102
103     this->visual_screen_frame_top.setOutlineThickness(2);
104     this->visual_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
105
106     this->visual_screen_frame_top.move(0, -2);
107
108
109     // 2. left framing
110     this->visual_screen_frame_left.setPointCount(n_points);
111
112     this->visual_screen_frame_left.setPoint(
113         0,
114         sf::Vector2f(this->position_x - 350, this->position_y + 50)
115     );
116     this->visual_screen_frame_left.setPoint(
117         1,
118         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
119     );
120     this->visual_screen_frame_left.setPoint(
121         2,
122         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
123     );
124     this->visual_screen_frame_left.setPoint(
125         3,
126         sf::Vector2f(this->position_x - 350, this->position_y + 350)
127     );
128
129     this->visual_screen_frame_left.setFillColor(VISUAL_SCREEN_FRAME_GREY);
130
131     this->visual_screen_frame_left.setOutlineThickness(2);
132     this->visual_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
133
134     this->visual_screen_frame_left.move(-2, 0);
135
136
137     // 3. bottom framing
138     this->visual_screen_frame_bottom.setPointCount(n_points);
139
140     this->visual_screen_frame_bottom.setPoint(
141         0,
142         sf::Vector2f(this->position_x - 350, this->position_y + 350)
143     );
144     this->visual_screen_frame_bottom.setPoint(
145         1,
146         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
147     );
148     this->visual_screen_frame_bottom.setPoint(
149         2,
150         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
151     );
152     this->visual_screen_frame_bottom.setPoint(
153         3,
154         sf::Vector2f(this->position_x - 50, this->position_y + 350)
155     );
156 }
```



```

156
157     this->visual_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
158
159     this->visual_screen_frame_bottom.setOutlineThickness(2);
160     this->visual_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
161
162     this->visual_screen_frame_bottom.move(0, 2);
163
164
165     // 4. right framing
166     this->visual_screen_frame_right.setPointCount(n_points);
167
168     this->visual_screen_frame_right.setPoint(
169         0,
170         sf::Vector2f(this->position_x - 50, this->position_y + 350)
171     );
172     this->visual_screen_frame_right.setPoint(
173         1,
174         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
175     );
176     this->visual_screen_frame_right.setPoint(
177         2,
178         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
179     );
180     this->visual_screen_frame_right.setPoint(
181         3,
182         sf::Vector2f(this->position_x - 50, this->position_y + 50)
183     );
184
185     this->visual_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
186
187     this->visual_screen_frame_right.setOutlineThickness(2);
188     this->visual_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
189
190     this->visual_screen_frame_right.move(2, 0);
191
192     return;
193 } /* __setUpVisualScreenFrame() */

```

3.2.3.15 draw()

```

void ContextMenu::draw (
    void )

```

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

```

932 {
933     // 1. menu frame
934     this->render_window_ptr->draw(this->menu_frame);
935
936     // 2. visual screen
937     this->render_window_ptr->draw(this->visual_screen);
938     this->__drawVisualScreenFrame();
939
940     // 3. console screen
941     this->render_window_ptr->draw(this->console_screen);
942     this->__drawConsoleScreenFrame();
943     this->__drawConsoleText();
944
945     this->frame++;
946     return;
947 } /* draw() */

```

3.2.3.16 processEvent()

```

void ContextMenu::processEvent (
    void )

```

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

```

830 {

```

```

831     if (this->event_ptr->type == sf::Event::KeyPressed) {
832         this->__handleKeyPressEvents();
833     }
834
835     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
836         this->__handleMouseButtonEvents();
837     }
838
839     return;
840 } /* processEvent() */

```

3.2.3.17 processMessage()

```

void ContextMenu::processMessage (
    void )

```

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

```

855 {
856     switch (this->console_state) {
857         case (ConsoleState :: TILE): {
858             // process no tile selected
859             if (not this->message_hub_ptr->isEmpty(NO_TILE_SELECTED_CHANNEL)) {
860                 Message no_tile_selected_message = this->message_hub_ptr->receiveMessage(
861                     NO_TILE_SELECTED_CHANNEL
862                 );
863
864                 if (no_tile_selected_message.subject == "no tile selected") {
865                     this->__setConsoleState(ConsoleState :: READY);
866
867                     std::cout << "No tile selected message received by " << this <<
868                         std::endl;
869                     this->message_hub_ptr->popMessage(NO_TILE_SELECTED_CHANNEL);
870                 }
871             }
872
873             // process tile state
874             if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
875                 Message tile_state_message = this->message_hub_ptr->receiveMessage(
876                     TILE_STATE_CHANNEL
877                 );
878
879                 if (tile_state_message.subject == "tile state") {
880                     this->console_string = tile_state_message.string_payload;
881
882                     std::cout << "Tile state message received by " << this << std::endl;
883                     this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
884                 }
885             }
886
887             // process tile selected (subsequent left clicks causing program to hang)
888             if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
889                 this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
890             }
891
892             break;
893         }
894
895         default: {
896             // process tile selected
897             if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
898                 Message tile_selected_message = this->message_hub_ptr->receiveMessage(
899                     TILE_SELECTED_CHANNEL
900                 );
901
902                 if (tile_selected_message.subject == "tile selected") {
903                     this->__setConsoleState(ConsoleState :: TILE);
904
905                     std::cout << "Tile selected message received by " << this <<
906                         std::endl;
907                     this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
908                 }
909             }
910
911             break;
912         }
913     }
914
915     return;
916 } /* processMessage() */

```

3.2.4 Member Data Documentation

3.2.4.1 assets_manager_ptr

`AssetsManager*` ContextMenu::assets_manager_ptr [private]

A pointer to the assets manager.

3.2.4.2 console_screen

`sf::RectangleShape` ContextMenu::console_screen

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

3.2.4.3 console_screen_frame_bottom

`sf::ConvexShape` ContextMenu::console_screen_frame_bottom

The bottom framing of the console screen.

3.2.4.4 console_screen_frame_left

`sf::ConvexShape` ContextMenu::console_screen_frame_left

The left framing of the console screen.

3.2.4.5 console_screen_frame_right

`sf::ConvexShape` ContextMenu::console_screen_frame_right

The right framing of the console screen.

3.2.4.6 console_screen_frame_top

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

The top framing of the console screen.

3.2.4.7 console_state

```
ConsoleState ContextMenu::console_state
```

The current state of the console screen.

3.2.4.8 console_string

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

The string to be printed to the console screen.

3.2.4.9 event_ptr

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

A pointer to the event class.

3.2.4.10 frame

```
int ContextMenu::frame
```

The current frame of this object.

3.2.4.11 game_menu_up

```
bool ContextMenu::game_menu_up
```

Indicates whether or not the game menu is up.

3.2.4.12 menu_frame

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

The frame of the context menu.

3.2.4.13 message_hub_ptr

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

A pointer to the message hub.

3.2.4.14 position_x

```
double ContextMenu::position_x
```

The position of the object.

3.2.4.15 position_y

```
double ContextMenu::position_y
```

The position of the object.

3.2.4.16 render_window_ptr

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

A pointer to the render window.

3.2.4.17 visual_screen

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

The context menu screen for visuals.

3.2.4.18 visual_screen_frame_bottom

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

The bottom framing of the visual screen.

3.2.4.19 visual_screen_frame_left

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

The left framing of the visual screen.

3.2.4.20 visual_screen_frame_right

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

The right framing of the visual screen.

3.2.4.21 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:

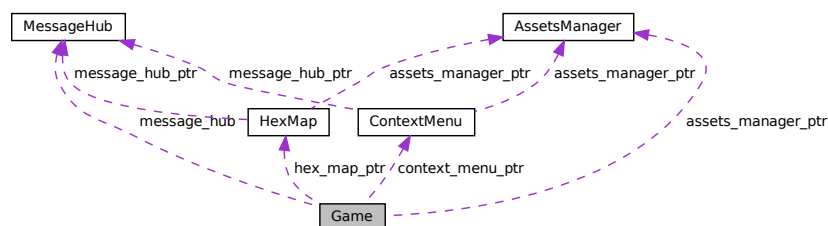
- header/[ContextMenu.h](#)
- source/[ContextMenu.cpp](#)

3.3 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

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

Private Member Functions

- void [__toggleFrameClockOverlay](#) (void)
Helper method to toggle frame clock overlay.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__processEvent](#) (void)
Helper method to process [Game](#). To be called once per event.
- void [__processMessage](#) (void)
Helper method to process [Game](#). To be called once per message.
- void [__sendGameStateMessage](#) (void)
Helper method to format and send a game state message.
- void [__insufficientCreditsAlarm](#) (void)
Helper method to sound and display and insufficient credits alarm.
- void [__drawFrameClockOverlay](#) (void)
Helper method to draw frame clock overlay.
- void [__drawHUD](#) (void)
Helper method to heads-up display (HUD).
- void [__draw](#) (void)
Helper method to draw game to the render window. To be called once per frame.

Private Attributes

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

3.3.1 Detailed Description

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

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Game()

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

Constructor for the `Game` class.

```
502 {
503     // 1. set attributes
504
505     // 1.1. private
506     this->render_window_ptr = render_window_ptr;
507
508     this->assets_manager_ptr = assets_manager_ptr;
509
510     // 1.2. public
511     this->quit_game = false;
512     this->game_loop_broken = false;
513     this->show_frame_clock_overlay = false;
514
515     this->frame = 0;
516     this->time_since_start_s = 0;
517
518     double seconds_since_epoch = time(NULL);
519     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
520
521     this->year = 1970 + (int)years_since_epoch;
522     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
523
524     this->population = 0;
525     this->credits = 250;
526     this->demand_MWh = 0;
527     this->cumulative_emissions_tonnes = 0;
528
529     this->hex_map_ptr = new HexMap(
530         6,
531         &(this->event),
532         this->render_window_ptr,
533         this->assets_manager_ptr,
534         &(this->message_hub)
535     );
536
537     this->context_menu_ptr = new ContextMenu(
538         &(this->event),
539         this->render_window_ptr,
540         this->assets_manager_ptr,
541         &(this->message_hub)
542     );
543
544     // 2. add message channel(s)
545     this->message_hub.addChannel(GAME_CHANNEL);
546     this->message_hub.addChannel(GAME_STATE_CHANNEL);
547
548     std::cout << "Game constructed at " << this << std::endl;
549
550     return;
551 } /* Game() */
```

3.3.2.2 ~Game()

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

Destructor for the `Game` class.

```
628 {
629     // 1. clean up attributes
630     delete this->hex_map_ptr;
631     delete this->context_menu_ptr;
632
633     std::cout << "Game at " << this << " destroyed" << std::endl;
634
635     return;
636 } /* ~Game() */
```

3.3.3 Member Function Documentation

3.3.3.1 `__draw()`

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

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

```
469 {
470     this->__drawHUD();
471
472     if (this->show_frame_clock_overlay) {
473         this->__drawFrameClockOverlay();
474     }
475
476     return;
477 } /* draw() */
```

3.3.3.2 `__drawFrameClockOverlay()`

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

Helper method to draw frame clock overlay.

```
361 {
362     std::string frame_clock_string = "FRAME: ";
363     frame_clock_string += std::to_string(this->frame);
364     frame_clock_string += "\nTIME SINCE START [s]: ";
365     frame_clock_string += std::to_string(this->time_since_start_s);
366
367     sf::Text frame_clock_text(
368         frame_clock_string,
369         *(this->assets_manager_ptr->getFont("DroidSansMono")),
370         16
371     );
372
373     sf::RectangleShape frame_clock_backing(
374         sf::Vector2f(
375             1.02 * frame_clock_text.getLocalBounds().width,
376             1.20 * frame_clock_text.getLocalBounds().height
377         )
378     );
379     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
380
381     this->render_window_ptr->draw(frame_clock_backing);
382     this->render_window_ptr->draw(frame_clock_text);
383
384     return;
385 } /* __drawFrameClockOverlay() */
```

3.3.3.3 __drawHUD()

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

Helper method to heads-up display (HUD).

```
400 {
401     // 1. first line
402     std::string HUD_string = "YEAR: ";
403     HUD_string += std::to_string(this->year);
404
405     HUD_string += "    MONTH: ";
406     HUD_string += std::to_string(this->month);
407
408     HUD_string += "    POPULATION: ";
409     HUD_string += std::to_string(this->population);
410
411     HUD_string += "    CREDITS: ";
412     HUD_string += std::to_string(this->credits);
413     HUD_string += " K";
414
415     HUD_string += "    CURRENT DEMAND: ";
416     HUD_string += std::to_string(this->demand_MWh);
417     HUD_string += " MWh";
418
419     sf::Text HUD_text(
420         HUD_string,
421         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
422         16
423     );
424
425     HUD_text.setPosition(
426         (800 - HUD_text.getLocalBounds().width) / 2,
427         8
428     );
429
430     HUD_text.setFillColor(MONOCROME_TEXT_GREEN);
431
432     this->render_window_ptr->draw(HUD_text);
433
434
435     // 2. second line
436     HUD_string = "CUMULATIVE EMISSIONS: ";
437     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
438     HUD_string += " tonnes (CO2e)";
439
440     HUD_string += "    LIFETIME LIMIT: ";
441     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
442     HUD_string += " tonnes (CO2e)";
443
444     HUD_text.setString(HUD_string);
445
446     HUD_text.setPosition(
447         (800 - HUD_text.getLocalBounds().width) / 2,
448         35
449     );
450
451     this->render_window_ptr->draw(HUD_text);
452
453     return;
454 } /* __drawHUD() */
```

3.3.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
59 {
60     switch (this->event.key.code) {
61         case (sf::Keyboard::Tilde): {
62             this->__toggleFrameClockOverlay();
63
64             break;
65         }
66     }
```

```

66
67
68         case (sf::Keyboard::Tab): {
69             this->hex_map_ptr->toggleResourceOverlay();
70
71             break;
72         }
73
74
75         default: {
76             // do nothing!
77
78             break;
79         }
80     }
81
82     return;
83 } /* __handleKeyPressEvents() */

```

3.3.3.5 __handleMouseButtonEvents()

```

void Game::__handleMouseButtonEvents (
    void ) [private]

```

Helper method to handle mouse button events.

```

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

```

3.3.3.6 __insufficientCreditsAlarm()

```

void Game::__insufficientCreditsAlarm (
    void ) [private]

```

Helper method to sound and display and insufficient credits alarm.

```

266 {
267     // 1. sound buzzer
268     this->assets_manager_ptr->getSound("insufficient credits")->play();
269
270     // 2. construct alarm text and backing rectangle
271     sf::Text insufficient_credits_text(
272         "INSUFFICIENT CREDITS",
273         (* (this->assets_manager_ptr->getFont("DroidSansMono"))),
274         32
275     );
276
277     insufficient_credits_text.setOrigin(

```

```

278         insufficient_credits_text.getLocalBounds().width / 2,
279         insufficient_credits_text.getLocalBounds().height / 2
280     );
281
282     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
283
284     sf::RectangleShape backing_rectangle(
285         sf::Vector2f(
286             1.1 * insufficient_credits_text.getLocalBounds().width,
287             1.5 * insufficient_credits_text.getLocalBounds().height
288         )
289     );
290
291     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
292
293     backing_rectangle.setOrigin(
294         backing_rectangle.getLocalBounds().width / 2,
295         backing_rectangle.getLocalBounds().height / 2
296     );
297
298     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
299
300     // 3. blocking display loop (~3 seconds)
301     bool red_flag = true;
302     int alarm_frame = 0;
303     double time_since_alarm_s = 0;
304
305     sf::Clock alarm_clock;
306
307     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
308         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
309
310         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
311             this->render_window_ptr->clear();
312
313             this->hex_map_ptr->draw();
314             this->context_menu_ptr->draw();
315             this->__draw();
316
317             if (alarm_frame % 20 == 0) {
318                 if (red_flag) {
319                     red_flag = false;
320                 }
321
322                 else {
323                     red_flag = true;
324                 }
325             }
326
327             if (red_flag) {
328                 insufficient_credits_text.setFillColor(MONOCROME_TEXT_RED);
329             }
330
331             else {
332                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
333             }
334
335             this->render_window_ptr->draw(backing_rectangle);
336             this->render_window_ptr->draw(insufficient_credits_text);
337
338             this->render_window_ptr->display();
339
340             alarm_frame++;
341             this->frame++;
342         }
343     }
344
345     return;
346 } /* __insufficientCreditsAlarm( */

```

3.3.3.7 __processEvent()

```

void Game::__processEvent (
    void ) [private]

```

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

```

138 {
139     if (this->event.type == sf::Event::Closed) {

```

```

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

```

3.3.3.8 __processMessage()

```

void Game::__processMessage (
    void ) [private]

```

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

```

201 {
202     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
203         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
204
205         if (game_channel_message.subject == "quit game") {
206             this->quit_game = true;
207             this->game_loop_broken = true;
208
209             std::cout << "Quit game message received by " << this << std::endl;
210             this->message_hub.popMessage(GAME_CHANNEL);
211         }
212
213         if (game_channel_message.subject == "restart game") {
214             this->game_loop_broken = true;
215
216             std::cout << "Restart game message received by " << this << std::endl;
217             this->message_hub.popMessage(GAME_CHANNEL);
218         }
219
220         if (game_channel_message.subject == "state request") {
221             std::cout << "Game state request message received by " << this << std::endl;
222
223             this->__sendGameStateMessage();
224             this->message_hub.popMessage(GAME_CHANNEL);
225         }
226
227         if (game_channel_message.subject == "credits spent") {
228             this->credits -= game_channel_message.int_payload["credits spent"];
229
230             std::cout << "Credits spent message (" <<
231                 game_channel_message.int_payload["credits spent"] << ") received by "
232                 << this << std::endl;
233
234             std::cout << "Current credits (Game): " << this->credits << " K" <<
235                 std::endl;
236
237             this->message_hub.popMessage(GAME_CHANNEL);
238         }
239
240         if (game_channel_message.subject == "insufficient credits") {
241             std::cout << "Insufficient credits message received by " << this <<
242                 std::endl;
243
244             this->__insufficientCreditsAlarm();
245
246             this->message_hub.popMessage(GAME_CHANNEL);
247         }
248     }
249
250     return;
251 } /* __processMessage() */

```

3.3.3.9 __sendGameStateMessage()

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

Helper method to format and send a game state message.

```
168 {
169     Message game_state_message;
170
171     game_state_message.channel = GAME_STATE_CHANNEL;
172     game_state_message.subject = "game state";
173
174     game_state_message.int_payload["year"] = this->year;
175     game_state_message.int_payload["month"] = this->month;
176     game_state_message.int_payload["population"] = this->population;
177     game_state_message.int_payload["credits"] = this->credits;
178     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
179     game_state_message.int_payload["cumulative_emissions_tonnes"] =
180         this->cumulative_emissions_tonnes;
181
182     this->message_hub.sendMessage(game_state_message);
183
184     std::cout << "Game state message sent by " << this << std::endl;
185     return;
186 } /* __sendGameStateMessage() */
```

3.3.3.10 __toggleFrameClockOverlay()

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

Helper method to toggle frame clock overlay.

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

3.3.3.11 run()

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

Method to run game (defines game loop).

Returns

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

```

569 {
570     // 1. play brand animation
571     //...
572
573     // 2. show splash screen
574     //...
575
576     // 3. start game loop
577     while (not this->game_loop_broken) {
578         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
579
580         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
581             // 6.1. process events
582             while (this->render_window_ptr->pollEvent(this->event)) {
583                 this->hex_map_ptr->processEvent();
584                 this->context_menu_ptr->processEvent();
585                 this->__processEvent();
586             }
587
588
589             // 6.2. process messages
590             while (this->message_hub.hasTraffic()) {
591                 this->hex_map_ptr->processMessage();
592                 this->context_menu_ptr->processMessage();
593                 this->__processMessage();
594             }
595
596
597             // 6.3. draw frame
598             this->render_window_ptr->clear();
599
600             this->hex_map_ptr->draw();
601             this->context_menu_ptr->draw();
602             this->__draw();
603
604             this->render_window_ptr->display();
605
606
607             // 6.4. increment frame
608             this->frame++;
609         }
610     }
611
612     return this->quit_game;
613 } /* run() */

```

3.3.4 Member Data Documentation

3.3.4.1 assets_manager_ptr

`AssetsManager* Game::assets_manager_ptr [private]`

A pointer to the assets manager.

3.3.4.2 clock

`sf::Clock Game::clock`

The game clock.

3.3.4.3 context_menu_ptr

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

Pointer to the context menu.

3.3.4.4 credits

```
int Game::credits
```

Current balance of credits.

3.3.4.5 cumulative_emissions_tonnes

```
int Game::cumulative_emissions_tonnes
```

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

3.3.4.6 demand_MWh

```
int Game::demand_MWh
```

Current energy demand [MWh].

3.3.4.7 event

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

The game events class.

3.3.4.8 frame

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

The current frame of the game.

3.3.4.9 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

3.3.4.10 hex_map_ptr

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

Pointer to the hex map (defines game world).

3.3.4.11 message_hub

```
MessageHub Game::message_hub
```

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

3.3.4.12 month

```
int Game::month
```

Current game month.

3.3.4.13 population

```
int Game::population
```

Current population.

3.3.4.14 quit_game

```
bool Game::quit_game
```

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

3.3.4.15 render_window_ptr

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

A pointer to the render window.

3.3.4.16 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

3.3.4.17 time_since_start_s

```
double Game::time_since_start_s
```

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

3.3.4.18 year

```
int Game::year
```

Current game year.

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

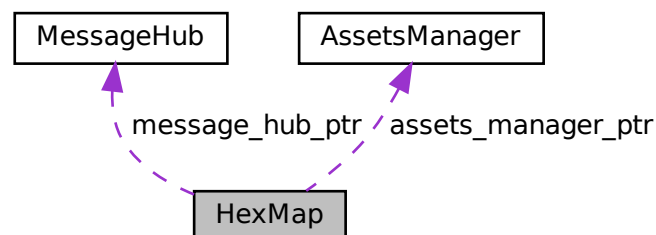
- header/[Game.h](#)
- source/[Game.cpp](#)

3.4 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 [tile_selected](#)
A boolean which indicates if a tile is currently selected.
- int [n_layers](#)
The number of layers in the hex map.
- int [n_tiles](#)
The number of tiles in the hex map.
- int [frame](#)
The current frame of this object.
- double [position_x](#)
The x position of the hex map's origin (i.e. central) tile.
- double [position_y](#)
The y position of the hex map's origin (i.e. central) tile.
- sf::RectangleShape [glass_screen](#)
To give the effect of an old glass screen over the hex map.
- std::vector< double > [tile_position_x_vec](#)
A vector of tile x positions.
- std::vector< double > [tile_position_y_vec](#)
A vector of tile y position.
- std::vector< [HexTile](#) * > [border_tiles_vec](#)
A vector of pointers to the border tiles.
- std::map< double, std::map< double, [HexTile](#) * > > [hex_map](#)
A position-indexed, nested map of hex tiles.
- std::vector< [HexTile](#) * > [hex_draw_order_vec](#)
A vector of hex tiles, in drawing order.

Private Member Functions

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

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.

3.4.1 Detailed Description

A class which defines a hex map of hex tiles.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 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.

```
1057 {
1058     // 1. set attributes
1059
1060     // 1.1. private
1061     this->event_ptr = event_ptr;
1062     this->render_window_ptr = render_window_ptr;
1063
1064     this->assets_manager_ptr = assets_manager_ptr;
1065     this->message_hub_ptr = message_hub_ptr;
1066
1067     // 1.2. public
1068     this->tile_selected = false;
1069
1070     this->frame = 0;
1071
1072     this->n_layers = n_layers;
1073     if (this->n_layers < 0) {
1074         this->n_layers = 0;
1075     }
1076
1077     this->position_x = 400;
1078     this->position_y = 400;
1079
1080     // 2. assemble n layer hex map
1081     this->__assembleHexMap();
1082
1083     // 3. set up and position drawable attributes
1084     this->__setUpGlassScreen();
1085
1086     // 4. add message channel(s)
1087     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1088     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1089     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1090
1091     std::cout << "HexMap constructed at " << this << std::endl;
1092
1093     return;
1094 } /* HexMap(), intended */
```

3.4.2.2 ~HexMap()

```
HexMap::~HexMap (
    void )
```

Destructor for the [HexMap](#) class.

```
1346 {
1347     this->clear();
1348
1349     std::cout << "HexMap at " << this << " destroyed" << std::endl;
1350
1351     return;
1352 } /* ~HexMap() */
```

3.4.3 Member Function Documentation

3.4.3.1 __assembleHexMap()

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

Helper method to assemble the hex map.

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

3.4.3.2 __buildDrawOrderVector()

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

Helper method to build tile drawing order vector.

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

```

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

```

3.4.3.3 __enforceOceanContinuity()

```

void HexMap::__enforceOceanContinuity (
    void ) [private]

```

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

```

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

```



```

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

```

3.4.3.4 __getMajorityTileType()

```

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

```

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

Parameters

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

Returns

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

```

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

```

```

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

```

3.4.3.5 __getNeighboursVector()

```

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

```

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

Parameters

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

Returns

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

```

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

```

3.4.3.6 __getNoise()

```

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

```

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

Parameters

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

Returns

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

```

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

```

```

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

```

3.4.3.7 __getSelectedTile()

```

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

```

Helper method to get pointer to selected tile.

Returns

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

```

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

```

3.4.3.8 __getValidMapIndexPositions()

```

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

```

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

Parameters

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

Returns

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

```

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

```

3.4.3.9 __handleKeyPressEvents()

```

void HexMap::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

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

```

3.4.3.10 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

3.4.3.11 __isLakeTouchingOcean()

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

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

3.4.3.12 __layTiles()

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

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

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

```

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

```



```

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

```

3.4.3.13 __procedurallyGenerateTileResources()

```

void HexMap::__procedurallyGenerateTileResources (
    void ) [private]

```

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

```

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

```

3.4.3.14 __procedurallyGenerateTileTypes()

```

void HexMap::__procedurallyGenerateTileTypes (
    void ) [private]

```

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

```

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

```

```

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

```

3.4.3.15 __sendNoTileSelectedMessage()

```

void HexMap::__sendNoTileSelectedMessage (
    void ) [private]

```

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

```

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

```

3.4.3.16 __setUpGlassScreen()

```

void HexMap::__setUpGlassScreen (
    void ) [private]

```

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

```

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

```

3.4.3.17 __smoothTileTypes()

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

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

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

3.4.3.18 assess()

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

Method to assess the resource of the selected tile.

```
1109 {
1110     HexTile* selected_tile_ptr = this->__getSelectedTile();
1111     if (selected_tile_ptr != NULL) {
1112         selected_tile_ptr->assess();
1113     }
1114
1115     return;
1116 } /* assess() */
```

3.4.3.19 clear()

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

Method to clear the hex map.

```
1308 {
1309     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1310     std::map<double, HexTile*>::iterator hex_map_iter_y;
1311     for (
1312         hex_map_iter_x = this->hex_map.begin();
1313         hex_map_iter_x != this->hex_map.end();
1314         hex_map_iter_x++
1315     ) {
1316         for (
```

```

1317         hex_map_iter_y = hex_map_iter_x->second.begin();
1318         hex_map_iter_y != hex_map_iter_x->second.end();
1319         hex_map_iter_y++
1320     ) {
1321         delete hex_map_iter_y->second;
1322     }
1323 }
1324 this->hex_map.clear();
1325
1326 this->tile_position_x_vec.clear();
1327 this->tile_position_y_vec.clear();
1328 this->border_tiles_vec.clear();
1329
1330 return;
1331 } /* clear() */

```

3.4.3.20 draw()

```

void HexMap::draw (
    void )

```

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

```

1265 {
1266     // 1. draw background
1267     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1268     glass_screen_colour.a = 255;
1269     this->glass_screen.setFillColor(glass_screen_colour);
1270
1271     this->render_window_ptr->draw(this->glass_screen);
1272
1273     // 2. draw tiles in drawing order
1274     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1275         this->hex_draw_order_vec[i]->draw();
1276     }
1277
1278     // 3. redraw selected tile
1279     HexTile* selected_tile_ptr = this->__getSelectedTile();
1280     if (selected_tile_ptr != NULL) {
1281         selected_tile_ptr->draw();
1282     }
1283
1284     // 4. draw glass screen
1285     glass_screen_colour = this->glass_screen.getFillColor();
1286     glass_screen_colour.a = 40;
1287     this->glass_screen.setFillColor(glass_screen_colour);
1288
1289     this->render_window_ptr->draw(this->glass_screen);
1290
1291     this->frame++;
1292     return;
1293 } /* draw() */

```

3.4.3.21 processEvent()

```

void HexMap::processEvent (
    void )

```

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

```

1184 {
1185     // 1. process HexTile events
1186     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1187     std::map<double, HexTile*>::iterator hex_map_iter_y;
1188     for (
1189         hex_map_iter_x = this->hex_map.begin();
1190         hex_map_iter_x != this->hex_map.end();
1191         hex_map_iter_x++
1192     ) {
1193         for (
1194             hex_map_iter_y = hex_map_iter_x->second.begin();

```

```

1195         hex_map_iter_y != hex_map_iter_x->second.end();
1196         hex_map_iter_y++
1197     ) {
1198         hex_map_iter_y->second->processEvent();
1199     }
1200 }
1201
1202 // 2. process HexMap events
1203 if (this->event_ptr->type == sf::Event::KeyPressed) {
1204     this->__handleKeyPressEvents();
1205 }
1206
1207 if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1208     this->__handleMouseButtonEvents();
1209 }
1210
1211 return;
1212 } /* processEvent() */

```

3.4.3.22 processMessage()

```

void HexMap::processMessage (
    void )

```

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

```

1227 {
1228     // 1. process HexTile messages
1229     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1230     std::map<double, HexTile*>::iterator hex_map_iter_y;
1231     for (
1232         hex_map_iter_x = this->hex_map.begin();
1233         hex_map_iter_x != this->hex_map.end();
1234         hex_map_iter_x++
1235     ) {
1236         for (
1237             hex_map_iter_y = hex_map_iter_x->second.begin();
1238             hex_map_iter_y != hex_map_iter_x->second.end();
1239             hex_map_iter_y++
1240         ) {
1241             hex_map_iter_y->second->processMessage();
1242         }
1243     }
1244
1245     // 2. process HexMap messages
1246     //...
1247
1248     return;
1249 } /* processMessage() */

```

3.4.3.23 reroll()

```

void HexMap::reroll (
    void )

```

Method to re-roll the hex map.

```

1131 {
1132     this->clear();
1133     this->__assembleHexMap();
1134
1135     return;
1136 } /* reroll() */

```

3.4.3.24 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
1151 {
1152     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1153     std::map<double, HexTile*>::iterator hex_map_iter_y;
1154     for (
1155         hex_map_iter_x = this->hex_map.begin();
1156         hex_map_iter_x != this->hex_map.end();
1157         hex_map_iter_x++
1158     ) {
1159         for (
1160             hex_map_iter_y = hex_map_iter_x->second.begin();
1161             hex_map_iter_y != hex_map_iter_x->second.end();
1162             hex_map_iter_y++
1163         ) {
1164             hex_map_iter_y->second->toggleResourceOverlay();
1165         }
1166     }
1167     return;
1168 } /* toggleResourceOverlay() */
1169 }
```

3.4.4 Member Data Documentation

3.4.4.1 assets_manager_ptr

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

A pointer to the assets manager.

3.4.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

3.4.4.3 event_ptr

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

A pointer to the event class.

3.4.4.4 frame

```
int HexMap::frame
```

The current frame of this object.

3.4.4.5 glass_screen

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

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

3.4.4.6 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

3.4.4.7 hex_map

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

A position-indexed, nested map of hex tiles.

3.4.4.8 message_hub_ptr

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

A pointer to the message hub.

3.4.4.9 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

3.4.4.10 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

3.4.4.11 position_x

```
double HexMap::position_x
```

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

3.4.4.12 position_y

```
double HexMap::position_y
```

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

3.4.4.13 render_window_ptr

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

A pointer to the render window.

3.4.4.14 tile_position_x_vec

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

A vector of tile x positions.

3.4.4.15 tile_position_y_vec

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

A vector of tile y position.

3.4.4.16 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:

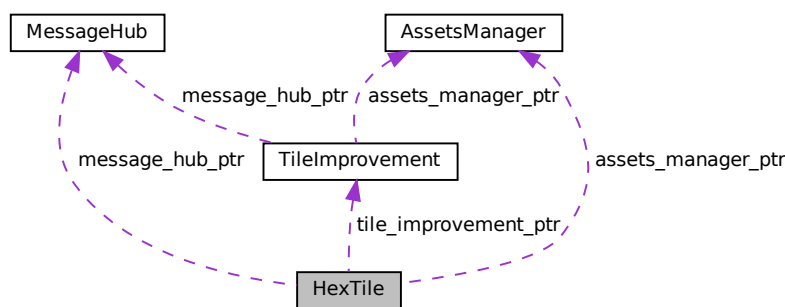
- header/[HexMap.h](#)
- source/[HexMap.cpp](#)

3.5 HexTile Class Reference

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

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

Collaboration diagram for HexTile:



Public Member Functions

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

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

Public Attributes

- [TileType](#) [tile_type](#)
- [TileResource](#) [tile_resource](#)
- bool [show_node](#)
A boolean which indicates whether or not to show the tile node.
- bool [show_resource](#)
A boolean which indicates whether or not to show resource value.
- bool [resource_assessed](#)
A boolean which indicates whether or not the resource has been assessed.
- bool [is_selected](#)
A boolean which indicates whether or not the tile is selected.
- bool [settlement_built](#)
A boolean which indicates if a settlement has been built yet or not.
- 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.
- int [frame](#)
The current frame of this object.
- int [credits](#)
The current balance of credits.
- double [position_x](#)
The x position of the tile.
- double [position_y](#)
The y position of the tile.
- double [major_radius](#)
The radius of the smallest bounding circle.
- double [minor_radius](#)
The radius of the largest inscribed circle.
- 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.

Private Member Functions

- void [__setUpNodeSprite](#) (void)
Helper method to set up node sprite.
- void [__setUpTileSprite](#) (void)
Helper method to set up tile sprite.
- void [__setUpSelectOutlineSprite](#) (void)
Helper method to set up select outline sprite.
- void [__setUpResourceChipSprite](#) (void)
Helper method to set up resource chip sprite.
- void [__setResourceText](#) (void)
Helper method to set up resource text.
- bool [__isClicked](#) (void)
Helper method to determine if tile was clicked on.
- void [__handleKeyPressEvents](#) (void)
Helper method to handle key press events.
- void [__handleMouseButtonEvents](#) (void)
Helper method to handle mouse button events.
- void [__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 [__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.

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.

3.5.1 Detailed Description

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

3.5.2 Constructor & Destructor Documentation

3.5.2.1 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.

```
763 {
764     // 1. set attributes
765
766     // 1.1. private
767     this->event_ptr = event_ptr;
768     this->render_window_ptr = render_window_ptr;
769
770     this->assets_manager_ptr = assets_manager_ptr;
771     this->message_hub_ptr = message_hub_ptr;
772
773     // 1.2. public
774     this->show_node = false;
775     this->show_resource = false;
776     this->resource_assessed = false;
777     this->is_selected = false;
778
779     this->has_improvement = false;
780     this->settlement_built = false;
781     this->tile_improvement_ptr = NULL;
782
783     this->frame = 0;
784     this->credits = 0;
785
786     this->position_x = position_x;
787     this->position_y = position_y;
788
789     this->major_radius = 32;
790     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
791
792     // 2. set up and position drawable attributes
793     this->__setUpNodeSprite();
```

```

794     this->__setUpTileSprite();
795     this->__setUpSelectOutlineSprite();
796     this->__setUpResourceChipSprite();
797     this->__setResourceText();
798
799     // 3. set tile type and resource (default to none type and average)
800     this->setTileType(TileType :: NONE_TYPE);
801     this->setTileResource(TileResource :: AVERAGE);
802
803     std::cout << "HexTile constructed at " << this << std::endl;
804
805     return;
806 } /* HexTile() */

```

3.5.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

1270 {
1271     if (this->tile_improvement_ptr != NULL) {
1272         delete this->tile_improvement_ptr;
1273     }
1274
1275     std::cout << "HexTile at " << this << " destroyed" << std::endl;
1276
1277     return;
1278 } /* ~HexTile() */

```

3.5.3 Member Function Documentation

3.5.3.1 __getTileCoordsSubstring()

```

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

```

Helper method to assemble and return tile coordinates substring.

Returns

Tile coordinates substring.

```

409 {
410     std::string coords_substring = "TILE COORDS: ";
411     coords_substring += std::to_string(int(this->position_x - 400));
412     coords_substring += ", ";
413     coords_substring += std::to_string(int(this->position_y - 400));
414     coords_substring += "\n";
415
416     return coords_substring;
417 } /* __getTileCoordsSubstring() */

```

3.5.3.2 __getTileImprovementSubstring()

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

Helper method to assemble and return the tile improvement substring.

Returns

Tile improvement substring.

```
536 {
537     std::string improvement_substring = "TILE IMPROVEMENT: ";
538
539     if (this->has_improvement) {
540         //...
541     }
542
543     else {
544         improvement_substring += "NONE\n";
545     }
546
547     return improvement_substring;
548 } /* __getTileImprovementSubstring() */
```

3.5.3.3 __getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

```
565 {
566     //          32 char x 17 line console "-----\n";
567     std::string options_substring = "      **** TILE OPTIONS **** \n";
568     options_substring += "      \n";
569
570     if (
571         (not this->has_improvement) and
572         (not this->settlement_built) and
573         (this->tile_type != TileType :: OCEAN) and
574         (this->tile_type != TileType :: LAKE)
575     ) {
576         options_substring += "[B]:  BUILD SETTLEMENT      ";
577     }
578
579     else if (not this->has_improvement) {
580         switch (this->tile_type) {
581             //...
582
583             default: {
584                 // do nothing!
585                 break;
586             }
587         }
588     }
589 }
590
591
592 return options_substring;
593 } /* __getTileOptionsString() */
```

3.5.3.4 __getTileResourceSubstring()

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

Helper method to assemble and return tile resource substring.

Returns

Tile resource substring.

```
498 {
499     std::string resource_substring = "TILE RESOURCE:      ";
500
501     if (this->resource_assessed) {
502         switch (this->tile_resource) {
503             //...
504
505             default: {
506                 resource_substring += "???\n";
507                 break;
508             }
509         }
510     }
511 }
512
513 else {
514     resource_substring += "[A]: ASSESS\n";
515 }
516
517 return resource_substring;
518 } /* __getTileResourceSubstring() */
519 }
```

3.5.3.5 __getTileTypeSubstring()

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

Helper method to assemble and return tile type substring.

Returns

Tile type substring.

```
434 {
435     std::string type_substring = "TILE TYPE:          ";
436
437     switch (this->tile_type) {
438         case (TileType :: FOREST): {
439             type_substring += "FOREST\n";
440             break;
441         }
442
443         case (TileType :: LAKE): {
444             type_substring += "LAKE\n";
445             break;
446         }
447
448         case (TileType :: MOUNTAINS): {
449             type_substring += "MOUNTAINS\n";
450             break;
451         }
452
453         case (TileType :: OCEAN): {
454
455
456
457
458
459     }
```

```

460         type_substring += "OCEAN\n";
461     }
462     break;
463 }
464
465 case (TileType :: PLAINS): {
466     type_substring += "PLAINS\n";
467 }
468 break;
469 }
470
471 default: {
472     type_substring += "???\n";
473 }
474 break;
475 }
476 }
477
478 return type_substring;
479 }
480 /* __getTileTypeSubstring() */
481 }

```

3.5.3.6 __handleKeyPressEvents()

```

void HexTile::__handleKeyPressEvents (
    void ) [private]

```

Helper method to handle key press events.

```

277 {
278     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
279         this->is_selected = false;
280     }
281
282     if (this->is_selected) {
283         switch (this->event_ptr->key.code) {
284             case (sf::Keyboard::A): {
285                 if (this->resource_assessed) {
286                     std::cout << "Cannot assess resource: already assessed" <<
287                         std::endl;
288                 }
289
290                 else if (this->credits < RESOURCE_ASSESSMENT_COST) {
291                     std::cout << "Cannot assess resource: insufficient credits (need "
292                         << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
293
294                     this->__sendInsufficientCreditsMessage();
295                 }
296
297                 else {
298                     this->assess();
299                     this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
300                     this->__sendGameStateRequest();
301                 }
302             }
303             break;
304         }
305
306         default: {
307             // do nothing!
308         }
309         break;
310     }
311 }
312 }
313
314 return;
315 }
316 /* __handleKeyPressEvents() */

```


3.5.3.7 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
331 {
332     switch (this->event_ptr->mouseButton.button) {
333         case (sf::Mouse::Left): {
334             if (this->__isClicked()) {
335                 std::cout << "Tile (" << this->position_x << ", " <<
336                     this->position_y << ") was selected" << std::endl;
337
338                 this->is_selected = true;
339
340                 this->__sendTileSelectedMessage();
341                 this->__sendTileStateMessage();
342                 this->__sendGameStateRequest();
343             }
344
345             else {
346                 this->is_selected = false;
347             }
348
349             break;
350         }
351
352         case (sf::Mouse::Right): {
353             this->is_selected = false;
354
355             break;
356         }
357
358         default: {
359             // do nothing!
360
361             break;
362         }
363     }
364 }
365
366 return;
367 }
368 /* __handleMouseButtonEvents() */
```

3.5.3.8 __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.

```
245 {
246     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
247
248     double mouse_x = mouse_position.x;
249     double mouse_y = mouse_position.y;
250
251     double distance = sqrt(
252         pow(this->position_x - mouse_x, 2) +
253         pow(this->position_y - mouse_y, 2)
254     );
255
256     if (distance < this->minor_radius) {
257         return true;
258     }
259     else {
260         return false;
261     }
262 }
263 /* __isClicked() */
```

3.5.3.9 __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.
----------------------	--

```
677 {
678     Message credits_spent_message;
679
680     credits_spent_message.channel = GAME_CHANNEL;
681     credits_spent_message.subject = "credits spent";
682
683     credits_spent_message.int_payload["credits spent"] = credits_spent;
684
685     this->message_hub_ptr->sendMessage(credits_spent_message);
686
687     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
688         << std::endl;
689     return;
690 } /* __sendCreditsSpentMessage() */
```

3.5.3.10 __sendGameStateRequest()

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

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

```
650 {
651     Message game_state_request;
652
653     game_state_request.channel = GAME_CHANNEL;
654     game_state_request.subject = "state request";
655
656     this->message_hub_ptr->sendMessage(game_state_request);
657
658     std::cout << "Game state request message sent by " << this << std::endl;
659     return;
660 } /* __sendGameStateRequest() */
```

3.5.3.11 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
705 {
706     Message insufficient_credits_message;
707
708     insufficient_credits_message.channel = GAME_CHANNEL;
709     insufficient_credits_message.subject = "insufficient credits";
710
711     this->message_hub_ptr->sendMessage(insufficient_credits_message);
712
713     std::cout << "Insufficient credits message sent by " << this << std::endl;
714
715     return;
716 } /* __sendInsufficientCreditsMessage() */
```

3.5.3.12 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
383 {
384     Message tile_selected_message;
385
386     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
387     tile_selected_message.subject = "tile selected";
388
389     this->message_hub_ptr->sendMessage(tile_selected_message);
390
391     return;
392 } /* __sendTileSelectedMessage() */
```

3.5.3.13 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
608 {
609     Message tile_state_message;
610
611     tile_state_message.channel = TILE_STATE_CHANNEL;
612     tile_state_message.subject = "tile state";
613
614
615     //          32 char x 17 line console "-----\n";
616     std::string string_payload          = "      **** TILE INFO ****      \n";
617     string_payload                     += "                                     \n";
618
619     string_payload                     += this->__getTileCoordsSubstring();
620     string_payload                     += "                                     \n";
621
622     string_payload                     += this->__getTileTypeSubstring();
623     string_payload                     += this->__getTileResourceSubstring();
624     string_payload                     += this->__getTileImprovementSubstring();
625     string_payload                     += "                                     \n";
626
627     string_payload                     += this->__getTileOptionsSubstring();
628
629
630     tile_state_message.string_payload = string_payload;
631
632     this->message_hub_ptr->sendMessage(tile_state_message);
633
634     std::cout << "Tile state message sent by " << this << std::endl;
635     return;
636 } /* __sendTileStateMessage() */
```

3.5.3.14 __setResourceText()

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

Helper method to set up resource text.

```
159 {
160     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
161
162     this->resource_text.setFill-color(sf::Color(0, 0, 0, 255));
163
164     if (this->resource_assessed) {
165         switch (this->tile_resource) {
```

```

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

```

3.5.3.15 __setUpNodeSprite()

```

void HexTile::__setUpNodeSprite (
    void ) [private]

```

Helper method to set up node sprite.

```

34 {
35     this->node_sprite.setRadius(4);
36
37     this->node_sprite.setOrigin(
38         this->node_sprite.getLocalBounds().width / 2,
39         this->node_sprite.getLocalBounds().height / 2
40     );
41 }

```

```

42     this->node_sprite.setPosition(this->position_x, this->position_y);
43
44     this->node_sprite.setFillColor(sf::Color(255, 0, 0, 255));
45
46     return;
47 } /* __setUpNodeSprite() */

```

3.5.3.16 __setUpResourceChipSprite()

```

void HexTile::__setUpResourceChipSprite (
    void ) [private]

```

Helper method to set up resource chip sprite.

```

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

```

3.5.3.17 __setUpSelectOutlineSprite()

```

void HexTile::__setUpSelectOutlineSprite (
    void ) [private]

```

Helper method to set up select outline sprite.

```

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

```

3.5.3.18 __setUpTileSprite()

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

Helper method to set up tile sprite.

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

3.5.3.19 assess()

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

Method to assess the tile's resource.

```
1125 {
1126     this->resource_assessed = true;
1127     this->__setResourceText();
1128     this->__sendTileStateMessage();
1129
1130     return;
1131 } /* assess() */
```

3.5.3.20 decorateTile()

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

Method to decorate tile.

```
1003 {
1004     switch (this->tile_type) {
1005         case (TileType :: FOREST): {
1006             this->tile_decoration_sprite.setTexture(
1007                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
1008             );
1009             break;
1010         }
1011
1012         case (TileType :: LAKE): {
1013             this->tile_decoration_sprite.setTexture(
1014                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
1015             );
1016             break;
1017         }
1018
1019     }
1020
1021     case (TileType :: MOUNTAINS): {
```

```

1022         this->tile_decoration_sprite.setTexture(
1023             *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
1024         );
1025
1026         break;
1027     }
1028
1029     case (TileType :: OCEAN): {
1030         this->tile_decoration_sprite.setTexture(
1031             *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
1032         );
1033
1034         break;
1035     }
1036
1037     case (TileType :: PLAINS): {
1038         this->tile_decoration_sprite.setTexture(
1039             *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
1040         );
1041
1042         break;
1043     }
1044
1045     default: {
1046         // do nothing!
1047
1048         break;
1049     }
1050 }
1051
1052
1053 if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
1054     this->tile_decoration_sprite.setOrigin(
1055         this->tile_decoration_sprite.getLocalBounds().width / 2,
1056         this->tile_decoration_sprite.getLocalBounds().height / 2
1057     );
1058
1059     this->tile_decoration_sprite.setPosition(
1060         this->position_x,
1061         this->position_y
1062     );
1063
1064     if ((double)rand() / RAND_MAX > 0.5) {
1065         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
1066     }
1067 }
1068
1069 else {
1070     this->tile_decoration_sprite.setOrigin(
1071         this->tile_decoration_sprite.getLocalBounds().width / 2,
1072         this->tile_decoration_sprite.getLocalBounds().height
1073     );
1074
1075     this->tile_decoration_sprite.setPosition(
1076         this->position_x,
1077         this->position_y + 12
1078     );
1079
1080     if ((double)rand() / RAND_MAX > 0.5) {
1081         this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
1082     }
1083 }
1084
1085 return;
1086 } /* decorateTile(void) */

```

3.5.3.21 draw()

```

void HexTile::draw (
    void )

```

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

```

1223 {
1224     // 1. draw hex
1225     this->render_window_ptr->draw(this->tile_sprite);
1226
1227     // 2. draw node
1228     if (this->show_node) {

```

```

1229         this->render_window_ptr->draw(this->node_sprite);
1230     }
1231
1232     // 3. draw tile decoration
1233     this->render_window_ptr->draw(this->tile_decoration_sprite);
1234
1235     // 4. draw resource
1236     if (this->show_resource) {
1237         this->render_window_ptr->draw(this->resource_chip_sprite);
1238         this->render_window_ptr->draw(this->resource_text);
1239     }
1240
1241     // 5. draw selection outline
1242     if (this->is_selected) {
1243         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
1244
1245         outline_colour.a =
1246             255 * pow(cos((M_PI * this->frame) / (1.5 * FRAMES_PER_SECOND)), 2);
1247
1248         this->select_outline_sprite.setOutlineColor(outline_colour);
1249
1250         this->render_window_ptr->draw(this->select_outline_sprite);
1251     }
1252
1253     this->frame++;
1254     return;
1255 } /* draw() */

```

3.5.3.22 processEvent()

```

void HexTile::processEvent (
    void )

```

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

```

1146 {
1147     // 1. process TileImprovement events
1148     if (this->tile_improvement_ptr != NULL) {
1149         this->tile_improvement_ptr->processEvent();
1150     }
1151
1152     // 2. process HexTile events
1153     if (this->event_ptr->type == sf::Event::KeyPressed) {
1154         this->__handleKeyPressEvents();
1155     }
1156
1157     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1158         this->__handleMouseButtonEvents();
1159     }
1160
1161     return;
1162 } /* processEvent() */

```

3.5.3.23 processMessage()

```

void HexTile::processMessage (
    void )

```

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

```

1177 {
1178     // 1. process TileImprovement messages
1179     if (this->tile_improvement_ptr != NULL) {
1180         this->tile_improvement_ptr->processMessage();
1181     }
1182
1183     // 2. process HexTile messages
1184     if (this->is_selected) {
1185         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
1186             Message game_state_message = this->message_hub_ptr->receiveMessage(
1187                 GAME_STATE_CHANNEL

```



```

1188         );
1189
1190         if (game_state_message.subject == "game state") {
1191             this->credits = game_state_message.int_payload["credits"];
1192
1193             if (this->tile_improvement_ptr != NULL) {
1194                 this->tile_improvement_ptr->credits = this->credits;
1195             }
1196
1197             std::cout << "Game state message received by " << this << std::endl;
1198             this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
1199         }
1200     }
1201
1202     std::cout << "Current credits (HexTile): " << this->credits << " K" <<
1203         std::endl;
1204 }
1205
1206 return;
1207 } /* processMessage() */

```

3.5.3.24 setTileResource() [1/2]

```

void HexTile::setTileResource (
    double input_value )

```

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

Parameters

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

```

952 {
953     // 1. check input
954     if (input_value < 0 or input_value > 1) {
955         std::string error_str = "ERROR HexTile::setTileResource() given input value is ";
956         error_str += "not in the closed interval [0, 1]";
957
958         #ifdef _WIN32
959             std::cout << error_str << std::endl;
960         #endif /* _WIN32 */
961
962         throw std::runtime_error(error_str);
963     }
964
965     // 2. convert input value to tile resource
966     TileResource tile_resource;
967
968     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
969         tile_resource = TileResource :: POOR;
970     }
971     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
972         tile_resource = TileResource :: BELOW_AVERAGE;
973     }
974     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
975         tile_resource = TileResource :: AVERAGE;
976     }
977     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
978         tile_resource = TileResource :: ABOVE_AVERAGE;
979     }
980     else {
981         tile_resource = TileResource :: GOOD;
982     }
983
984     // 3. call alternate method
985     this->setTileResource(tile_resource);
986
987     return;
988 } /* setTileResource(double) */

```

3.5.3.25 setTileResource() [2/2]

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

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

Parameters

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

```
930 {
931     this->tile_resource = tile_resource;
932     this->__setResourceText();
933
934     return;
935 } /* setTileResource(TileResource) */
```

3.5.3.26 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].
--------------------	--

```
880 {
881     // 1. check input
882     if (input_value < 0 or input_value > 1) {
883         std::string error_str = "ERROR HexTile::setTileType() given input value is ";
884         error_str += "not in the closed interval [0, 1]";
885
886         #ifdef _WIN32
887             std::cout << error_str << std::endl;
888         #endif /* _WIN32 */
889
890         throw std::runtime_error(error_str);
891     }
892
893     // 2. convert input value to tile type
894     TileType tile_type;
895
896     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
897         tile_type = TileType :: LAKE;
898     }
899     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
900         tile_type = TileType :: PLAINS;
901     }
902     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
903         tile_type = TileType :: FOREST;
904     }
905     else {
906         tile_type = TileType :: MOUNTAINS;
907     }
908
909     // 3. call alternate method
910     this->setTileType(tile_type);
911
912     return;
913 } /* setTileType(double) */
```

3.5.3.27 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.
------------------	---

```
821 {
822     this->tile_type = tile_type;
823
824     switch (this->tile_type) {
825         case (TileType :: FOREST): {
826             this->tile_sprite.setFillColor(FOREST_GREEN);
827             break;
828         }
829
830         case (TileType :: LAKE): {
831             this->tile_sprite.setFillColor(LAKE_BLUE);
832             break;
833         }
834
835         case (TileType :: MOUNTAINS): {
836             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
837             break;
838         }
839
840         case (TileType :: OCEAN): {
841             this->tile_sprite.setFillColor(OCEAN_BLUE);
842             break;
843         }
844
845         case (TileType :: PLAINS): {
846             this->tile_sprite.setFillColor(PLAINS_YELLOW);
847             break;
848         }
849
850         default: {
851             // do nothing!
852             break;
853         }
854     }
855
856     return;
857 }
858
859 /* setTileType(TileType) */
```

3.5.3.28 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
1101 {
1102     if (this->show_resource) {
1103         this->show_resource = false;
1104     }
1105     else {
1106         this->show_resource = true;
1107     }
1108
1109     return;
1110 }
1111 /* toggleResourceOverlay() */
```

3.5.4 Member Data Documentation

3.5.4.1 assets_manager_ptr

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

A pointer to the assets manager.

3.5.4.2 credits

```
int HexTile::credits
```

The current balance of credits.

3.5.4.3 event_ptr

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

A pointer to the event class.

3.5.4.4 frame

```
int HexTile::frame
```

The current frame of this object.

3.5.4.5 has_improvement

```
bool HexTile::has_improvement
```

A boolean which indicates if tile has improvement or not.

3.5.4.6 is_selected

```
bool HexTile::is_selected
```

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

3.5.4.7 major_radius

```
double HexTile::major_radius
```

The radius of the smallest bounding circle.

3.5.4.8 message_hub_ptr

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

A pointer to the message hub.

3.5.4.9 minor_radius

```
double HexTile::minor_radius
```

The radius of the largest inscribed circle.

3.5.4.10 node_sprite

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

A circle shape to mark the tile node.

3.5.4.11 position_x

```
double HexTile::position_x
```

The x position of the tile.

3.5.4.12 position_y

```
double HexTile::position_y
```

The y position of the tile.

3.5.4.13 render_window_ptr

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

A pointer to the render window.

3.5.4.14 resource_assessed

```
bool HexTile::resource_assessed
```

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

3.5.4.15 resource_chip_sprite

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

A circle shape which represents a resource chip.

3.5.4.16 resource_text

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

A text representation of the resource.

3.5.4.17 select_outline_sprite

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

A convex shape which outlines the tile when selected.

3.5.4.18 settlement_built

```
bool HexTile::settlement_built
```

A boolean which indicates if a settlement has been built yet or not.

3.5.4.19 show_node

```
bool HexTile::show_node
```

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

3.5.4.20 show_resource

```
bool HexTile::show_resource
```

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

3.5.4.21 tile_decoration_sprite

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

A tile decoration sprite.

3.5.4.22 tile_improvement_ptr

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

A pointer to the improvement for this tile.

3.5.4.23 tile_resource

```
TileResource HexTile::tile_resource
```

3.5.4.24 tile_sprite

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

A convex shape which represents the tile.

3.5.4.25 tile_type

```
TileType HexTile::tile_type
```

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

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

3.6 Message Struct Reference

A structure which defines a standard message format.

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

Public Attributes

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

3.6.1 Detailed Description

A structure which defines a standard message format.

3.6.2 Member Data Documentation

3.6.2.1 bool_payload

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

A boolean payload.

3.6.2.2 channel

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

A string identifying the appropriate channel for this message.

3.6.2.3 double_payload

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

A vector payload.

3.6.2.4 int_payload

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

A vector payload.

3.6.2.5 string_payload

```
std::string Message::string_payload = ""
```

A string payload.

3.6.2.6 subject

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

A string describing the message subject.

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

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

3.7 MessageHub Class Reference

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

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

Public Member Functions

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

Private Attributes

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

3.7.1 Detailed Description

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

3.7.2 Constructor & Destructor Documentation

3.7.2.1 MessageHub()

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

Constructor for the [MessageHub](#) class.

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

3.7.2.2 ~MessageHub()

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

Destructor for the [MessageHub](#) class.

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

3.7.3 Member Function Documentation

3.7.3.1 addChannel()

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

Method to add channel to message map.

Parameters

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

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

```

113
114     std::cout << "Channel " << channel << " added to message hub" << std::endl;
115
116     return;
117 } /* addChannel() */

```

3.7.3.2 clear()

```

void MessageHub::clear (
    void )

```

Method to clear the [MessageHub](#).

```

373 {
374
375     this->clearMessages();
376     this->message_map.clear();
377
378     return;
379 } /* clear() */

```

3.7.3.3 clearMessages()

```

void MessageHub::clearMessages (
    void )

```

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

```

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

```

3.7.3.4 hasTraffic()

```

bool MessageHub::hasTraffic (
    void )

```

Method to determine if there remains any message traffic.

```

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

```

3.7.3.5 isEmpty()

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

Method to check if channel is empty.

Parameters

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

Returns

A boolean indicating whether the channel is empty or not.

```

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

```

3.7.3.6 popMessage()

```

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

```

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

Parameters

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

```

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

```

```

328     // 3. pop message
329     this->message_map[channel].pop_front();
330
331     return;
332 } /* popMessage() */

```

3.7.3.7 receiveMessage()

```

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

```

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

Parameters

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

Returns

The first message in the given channel.

```

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

```

3.7.3.8 removeChannel()

```

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

```

Method to remove channel from message map.

Parameters

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

```

134 {
135     // 1. check if channel is in map (if not, throw error)
136     if (this->message_map.count(channel) <= 0) {
137         std::string error_str = "ERROR MessageHub::removeChannel() channel ";
138         error_str += channel;
139         error_str += " is not in message map";
140
141         #ifdef _WIN32
142             std::cout << error_str << std::endl;
143         #endif /* _WIN32 */
144
145         throw std::runtime_error(error_str);
146     }
147
148     // 2. remove channel from map
149     this->message_map[channel].clear();
150     this->message_map.erase(channel);
151
152     std::cout << "Channel " << channel << " removed from message hub" << std::endl;
153
154     return;
155 } /* removeChannel() */

```

3.7.3.9 sendMessage()

```

void MessageHub::sendMessage (
    Message message )

```

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

Parameters

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

```

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

```

3.7.4 Member Data Documentation

3.7.4.1 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:

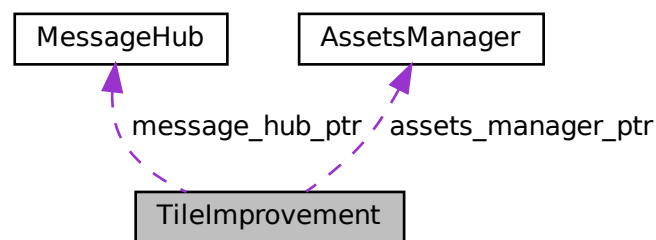
- header/ESC_core/[MessageHub.h](#)
- source/ESC_core/[MessageHub.cpp](#)

3.8 TileImprovement Class Reference

A base class for the tile improvement hierarchy.

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

Collaboration diagram for TileImprovement:



Public Member Functions

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

Public Attributes

- int [frame](#)
The current frame of this object.
- int [credits](#)
The current balance of credits.
- double [position_x](#)
The x position of the tile improvement.
- double [position_y](#)
The y position of the tile improvement.
- 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.

Private Member Functions

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

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.

3.8.1 Detailed Description

A base class for the tile improvement hierarchy.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 TileImprovement()

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

Constructor for the [TileImprovement](#) class.

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

Parameters

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

```

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

```

3.8.2.2 ~TileImprovement()

```

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

```

Destructor for the [TileImprovement](#) class.

```

233 {
234     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
235
236     return;
237 } /* ~TileImprovement() */

```

3.8.3 Member Function Documentation

3.8.3.1 __handleKeyPressEvents()

```

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

```

Helper method to handle key press events.

```

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

```

3.8.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
62 {
63     switch (this->event_ptr->mouseButton.button) {
64         case (sf::Mouse::Left): {
65             //...
66
67             break;
68         }
69
70         case (sf::Mouse::Right): {
71             //...
72
73             break;
74         }
75
76         default: {
77             // do nothing!
78
79             break;
80         }
81     }
82     return;
83 }
84
85 /* __handleMouseButtonEvents() */
86 }
```

3.8.3.3 draw()

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

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

```
213 {
214     //...
215
216     this->frame++;
217     return;
218 }
219 /* draw() */
```

3.8.3.4 processEvent()

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

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

```
168 {
169     if (this->event_ptr->type == sf::Event::KeyPressed) {
170         this->__handleKeyPressEvents();
171     }
172
173     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
174         this->__handleMouseButtonEvents();
175     }
176
177     return;
178 }
179 /* processEvent() */
```

3.8.3.5 processMessage()

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

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

```
193 {
194     //...
195
196     return;
197 } /* processMessage() */
```

3.8.4 Member Data Documentation

3.8.4.1 assets_manager_ptr

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

A pointer to the assets manager.

3.8.4.2 credits

```
int TileImprovement::credits
```

The current balance of credits.

3.8.4.3 event_ptr

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

A pointer to the event class.

3.8.4.4 frame

```
int TileImprovement::frame
```

The current frame of this object.

3.8.4.5 message_hub_ptr

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

A pointer to the message hub.

3.8.4.6 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

3.8.4.7 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

3.8.4.8 render_window_ptr

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

A pointer to the render window.

3.8.4.9 tile_improvement_sprite_animated

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

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

3.8.4.10 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

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

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

Chapter 4

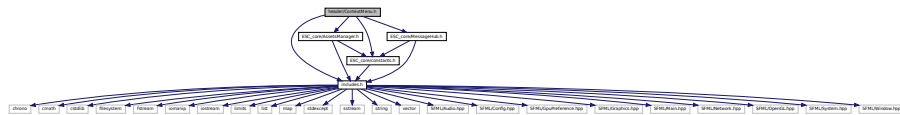
File Documentation

4.1 header/ContextMenu.h File Reference

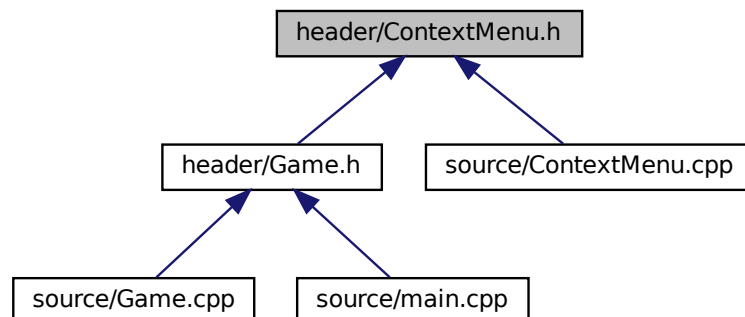
Header file for the [ContextMenu](#) class.

```
#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.

4.1.1 Detailed Description

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

4.1.2 Enumeration Type Documentation

4.1.2.1 ConsoleState

enum [ConsoleState](#)

An enumeration of the different console screen states.

Enumerator

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

```

34     {
35     NONE_STATE,
36     READY,
37     MENU,
38     TILE,
39     N_CONSOLE_STATES
40 };

```

4.2 header/ESC_core/AssetsManager.h File Reference

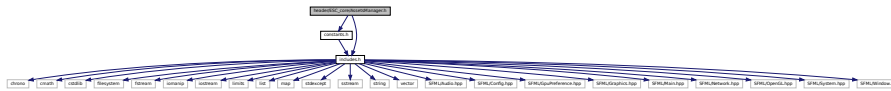
Header file for the [AssetsManager](#) class.

```

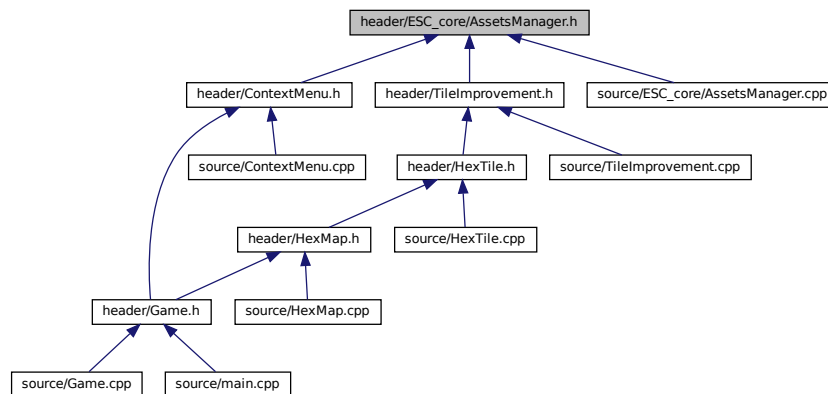
#include "constants.h"
#include "includes.h"

```


Include dependency graph for AssetsManager.h:



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



Classes

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

4.2.1 Detailed Description

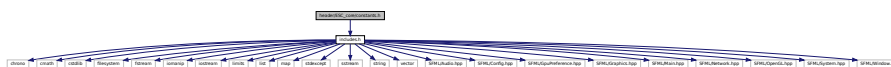
Header file for the [AssetsManager](#) class.

4.3 header/ESC_core/constants.h File Reference

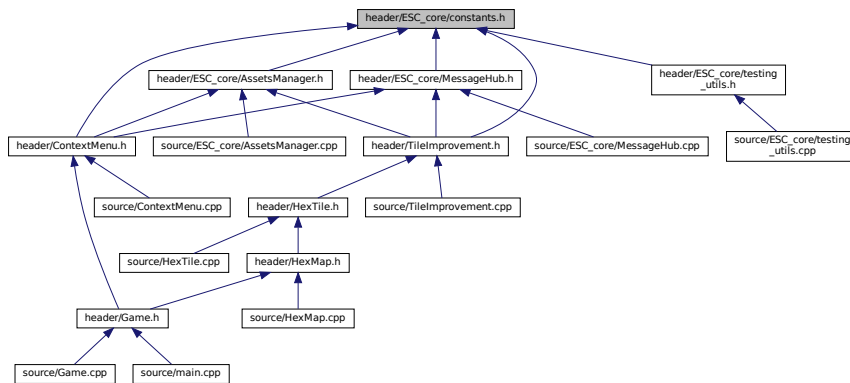
Header file for various constants.

```
#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 int `EMISSIONS_LIFETIME_LIMIT_TONNES` = 1500
The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.
- const int `RESOURCE_ASSESSMENT_COST` = 20
The cost of doing a resource assessment.
- const double `CO2E_KG_PER_LITRE_DIESEL` = 3.1596
The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.
- const std::string `GAME_CHANNEL` = "GAME CHANNEL"
A message channel for game messages.
- const std::string `GAME_STATE_CHANNEL` = "GAME STATE CHANNEL"
A message channel for game state messages.

4.3.1 Detailed Description

Header file for various constants.

4.3.2 Function Documentation

4.3.2.1 FOREST_GREEN()

```
const sf::Color FOREST_GREEN (
    34 ,
    139 ,
    34 )
```

The base colour of a forest tile.

4.3.2.2 LAKE_BLUE()

```
const sf::Color LAKE_BLUE (
    0 ,
    102 ,
    204 )
```

The base colour of a lake (water) tile.

4.3.2.3 MENU_FRAME_GREY()

```
const sf::Color MENU_FRAME_GREY (
    185 ,
    187 ,
    182 )
```

The base colour of the context menu frame.

4.3.2.4 MONOCHROME_SCREEN_BACKGROUND()

```
const sf::Color MONOCHROME_SCREEN_BACKGROUND (
    40 ,
    40 ,
    40 )
```

The base colour of old monochrome screens.

4.3.2.5 MONOCHROME_TEXT_AMBER()

```
const sf::Color MONOCHROME_TEXT_AMBER (
    255 ,
    176 ,
    0 )
```

The base colour of old monochrome text (amber).

4.3.2.6 MONOCHROME_TEXT_GREEN()

```
const sf::Color MONOCHROME_TEXT_GREEN (
    0 ,
    255 ,
    102 )
```

The base colour of old monochrome text (green).

4.3.2.7 MONOCHROME_TEXT_RED()

```
const sf::Color MONOCHROME_TEXT_RED (
    255 ,
    44 ,
    0 )
```

The base colour of old monochrome text (red).

4.3.2.8 MOUNTAINS_GREY()

```
const sf::Color MOUNTAINS_GREY (
    97 ,
    110 ,
    113 )
```

The base colour of a mountains tile.

4.3.2.9 OCEAN_BLUE()

```
const sf::Color OCEAN_BLUE (
    0 ,
    51 ,
    102 )
```

The base colour of an ocean (water) tile.

4.3.2.10 PLAINS_YELLOW()

```
const sf::Color PLAINS_YELLOW (
    245 ,
    222 ,
    133 )
```

The base colour of a plains tile.

4.3.2.11 RESOURCE_CHIP_GREY()

```
const sf::Color RESOURCE_CHIP_GREY (
    175 ,
    175 ,
    175 ,
    250 )
```

The base colour of the resource chip (backing).

4.3.2.12 VISUAL_SCREEN_FRAME_GREY()

```
const sf::Color VISUAL_SCREEN_FRAME_GREY (
    151 ,
    151 ,
    143 )
```

The base colour of the framing of the visual screen.

4.3.3 Variable Documentation

4.3.3.1 CO2E_KG_PER_LITRE_DIESEL

```
const double CO2E_KG_PER_LITRE_DIESEL = 3.1596
```

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

4.3.3.2 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.

4.3.3.3 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

4.3.3.4 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

4.3.3.5 GAME_CHANNEL

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

A message channel for game messages.

4.3.3.6 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

4.3.3.7 GAME_STATE_CHANNEL

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

A message channel for game state messages.

4.3.3.8 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

4.3.3.9 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

4.3.3.10 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

4.3.3.11 SECONDS_PER_FRAME

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

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

4.3.3.12 SECONDS_PER_MONTH

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

4.3.3.13 SECONDS_PER_YEAR

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

4.3.3.14 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).

4.3.3.15 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

4.3.3.16 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

4.3.3.17 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).

4.4 header/ESC_core/doxygen_cite.h File Reference

Header file which simply cites the doxygen tool.

4.4.1 Detailed Description

Header file which simply cites the doxygen tool.

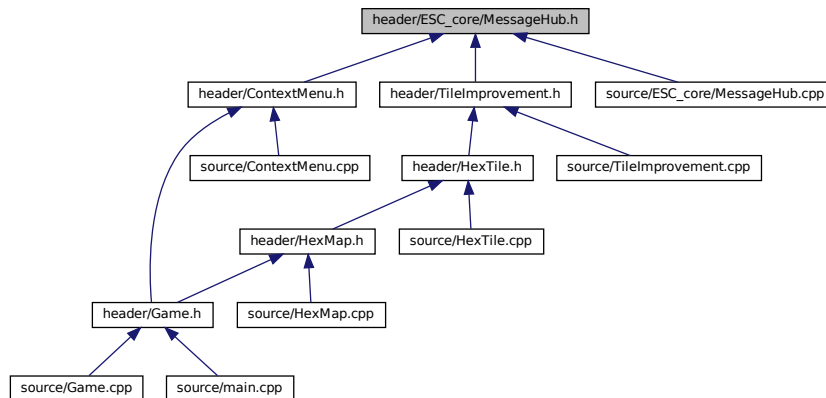
Ref: [van Heesch. \[2023\]](#)

4.5 header/ESC_core/includes.h File Reference

Header file for various includes.

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


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



Classes

- struct [Message](#)
A structure which defines a standard message format.
- class [MessageHub](#)
A class which acts as a central hub for inter-object message traffic.

4.6.1 Detailed Description

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

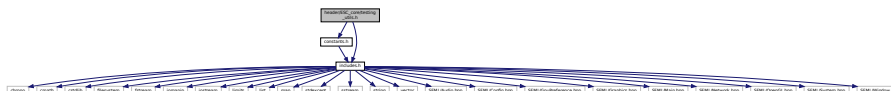
4.7 header/ESC_core/testing_utils.h File Reference

Header file for various testing utilities.

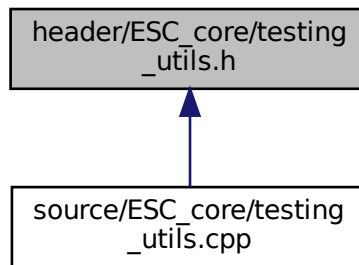
```
#include "constants.h"
```

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

Include dependency graph for `testing_utils.h`:



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



Functions

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

4.7.1 Detailed Description

Header file for various testing utilities.

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

4.7.2 Function Documentation

4.7.2.1 expectedErrorNotDetected()

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

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

Parameters

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

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

4.7.2.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {
83     std::cout << "\x1B[33m" << input_str << "\033[0m";
84     return;
85 } /* printGold() */
```

4.7.2.3 printGreen()

```
void printGreen (
    std::string input_str )
```

A function that sends green text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```

62 {
63     std::cout << "\xB[32m" << input_str << "\033[0m";
64     return;
65 } /* printGreen() */

```

4.7.2.4 printRed()

```

void printRed (
    std::string input_str )

```

A function that sends red text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```

102 {
103     std::cout << "\xB[31m" << input_str << "\033[0m";
104     return;
105 } /* printRed() */

```

4.7.2.5 testFloatEquals()

```

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

```

Tests for the equality of two floating point numbers *x* and *y* (to within `FLOAT_TOLERANCE`).

Parameters

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

```

136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";
150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif

```

```

156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

4.7.2.6 testGreaterThan()

```

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

```

Tests if $x > y$.

Parameters

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

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif
207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

4.7.2.7 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__").

```

240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);
260     return;
261 } /* testGreaterThanOrEqualTo() */

```

4.7.2.8 testLessThan()

```

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

```

Tests if $x < y$.

Parameters

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

```

291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);
311     return;

```



```
312 }    /* testLessThan() */
```

4.7.2.9 testLessThanOrEqualTo()

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

Tests if $x \leq y$.

Parameters

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

```
342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;
363 }    /* testLessThanOrEqualTo() */
```

4.7.2.10 testTruth()

```
void testTruth (
    bool statement,
    std::string file,
    int line )
```

Tests if the given statement is true.

Parameters

<i>statement</i>	The statement whose truth is to be tested ("1 == 0", for example).
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\n";
400     error_str += "Given statement is not true";
401
402     #ifdef _WIN32
403         std::cout << error_str << std::endl;
404     #endif
405
406     throw std::runtime_error(error_str);
407     return;
408 } /* testTruth() */

```

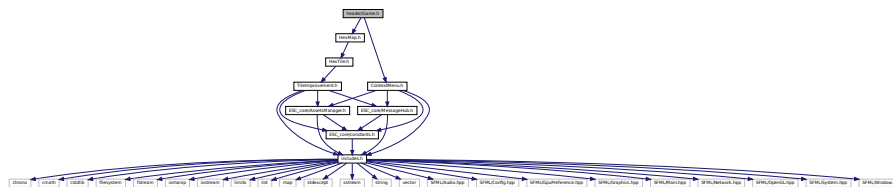
4.8 header/Game.h File Reference

```

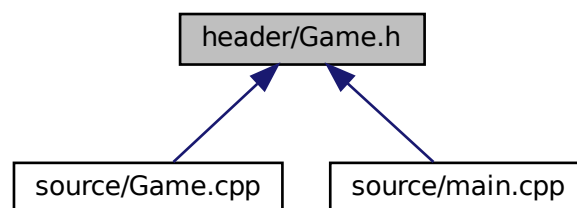
#include "HexMap.h"
#include "ContextMenu.h"

```

Include dependency graph for Game.h:



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



Classes

- class [Game](#)

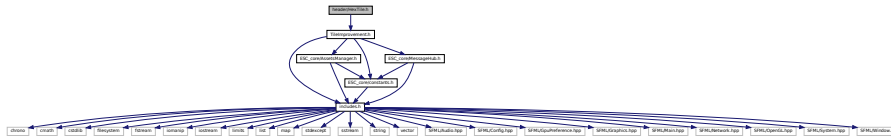
A class which acts as the central class for the game, by containing all other classes and implementing the game loop.

4.10 header/HexTile.h File Reference

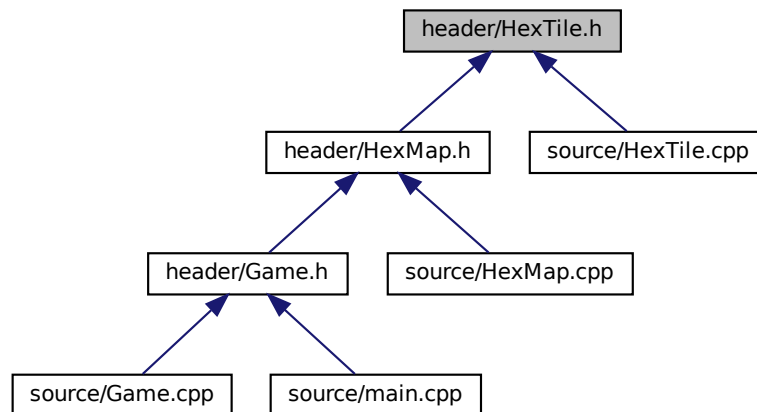
Header file for the [Game](#) class.

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

Include dependency graph for HexTile.h:



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



Classes

- class [HexTile](#)

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

Enumerations

- enum [TileType](#) {
[NONE_TYPE](#) , [FOREST](#) , [LAKE](#) , [MOUNTAINS](#) ,
[OCEAN](#) , [PLAINS](#) , [N_TILE_TYPES](#) }
An enumeration of the different tile types.
- enum [TileResource](#) {
[POOR](#) , [BELOW_AVERAGE](#) , [AVERAGE](#) , [ABOVE_AVERAGE](#) ,
[GOOD](#) , [N_TILE_RESOURCES](#) }
An enumeration of the different tile resource values.

4.10.1 Detailed Description

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

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

4.10.2 Enumeration Type Documentation

4.10.2.1 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.

```
48         {
49     POOR,
50     BELOW_AVERAGE,
51     AVERAGE,
52     ABOVE_AVERAGE,
53     GOOD,
54     N_TILE_RESOURCES
55 }; /* TileResource */
```

4.10.2.2 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.

```

31     {
32         NONE_TYPE,
33         FOREST,
34         LAKE,
35         MOUNTAINS,
36         OCEAN,
37         PLAINS,
38         N_TILE_TYPES
39     }; /* TileType */

```

4.11 header/TileImprovement.h File Reference

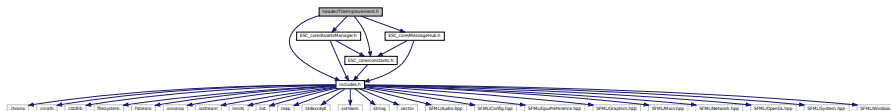
Header file for the [TileImprovement](#) class.

```

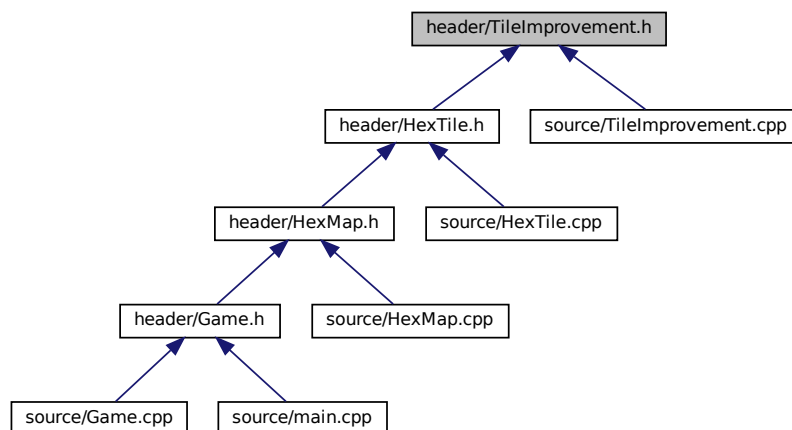
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"

```

Include dependency graph for TileImprovement.h:



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



Classes

- class [TileImprovement](#)

A base class for the tile improvement hierarchy.

Enumerations

- enum [TileImprovementType](#) {
[SETTLEMENT](#) , [SOLAR_PV](#) , [WIND_TURBINE](#) , [TIDAL_TURBINE](#) ,
[WAVE_ENERGY_CONVERTER](#) , [ENERGY_STORAGE_SYSTEM](#) , [N_TILE_IMPROVEMENT_TYPES](#) }

An enumeration of the different tile improvement types.

4.11.1 Detailed Description

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

4.11.2 Enumeration Type Documentation

4.11.2.1 TileImprovementType

enum [TileImprovementType](#)

An enumeration of the different tile improvement types.

Enumerator

SETTLEMENT	A settlement.
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.

```

34         {
35     SETTLEMENT,
36     SOLAR\_PV,
37     WIND\_TURBINE,
38     TIDAL\_TURBINE,
39     WAVE\_ENERGY\_CONVERTER,
40     ENERGY\_STORAGE\_SYSTEM,
41     N\_TILE\_IMPROVEMENT\_TYPES
42 }; /* TileImprovementType */

```

4.12 source/ContextMenu.cpp File Reference

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

4.14.1 Detailed Description

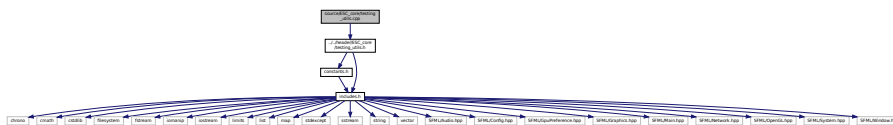
Implementation file for the `MessageHub` class.

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

4.15 source/ESC_core/testing_utils.cpp File Reference

Implementation file for various testing utilities.

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



Functions

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

4.15.1 Detailed Description

Implementation file for various testing utilities.

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

4.15.2 Function Documentation

4.15.2.1 expectedErrorNotDetected()

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

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

Parameters

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

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

4.15.2.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {
83     std::cout << "\x1B[33m" << input_str << "\033[0m";
84     return;
85 } /* printGold() */
```

4.15.2.3 printGreen()

```
void printGreen (
    std::string input_str )
```

A function that sends green text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```

62 {
63     std::cout << "\x1B[32m" << input_str << "\033[0m";
64     return;
65 } /* printGreen() */

```

4.15.2.4 printRed()

```

void printRed (
    std::string input_str )

```

A function that sends red text to std::cout.

Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```

102 {
103     std::cout << "\x1B[31m" << input_str << "\033[0m";
104     return;
105 } /* printRed() */

```

4.15.2.5 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__").

```

136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";

```

```

150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif
156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

4.15.2.6 testGreaterThan()

```

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

```

Tests if $x > y$.

Parameters

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

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif
207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

4.15.2.7 testGreaterThanOrEqualTo()

```

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

```

Tests if $x \geq y$.

Parameters

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

```

240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);
260     return;
261 } /* testGreaterThanOrEqualTo() */

```

4.15.2.8 testLessThan()

```

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

```

Tests if $x < y$.

Parameters

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

```

291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);

```

```

311     return;
312 } /* testLessThan() */

```

4.15.2.9 testLessThanOrEqualTo()

```

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

```

Tests if $x \leq y$.

Parameters

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

```

342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;
363 } /* testLessThanOrEqualTo() */

```

4.15.2.10 testTruth()

```

void testTruth (
    bool statement,
    std::string file,
    int line )

```

Tests if the given statement is true.

Parameters

<i>statement</i>	The statement whose truth is to be tested ("1 == 0", for example).
<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```

390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\t\n";
400     error_str += "Given statement is not true";
401
402     #ifdef _WIN32
403         std::cout << error_str << std::endl;
404     #endif
405
406     throw std::runtime_error(error_str);
407     return;
408 } /* testTruth() */

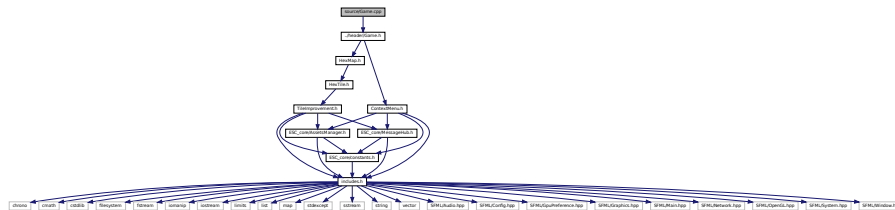
```

4.16 source/Game.cpp File Reference

Implementation file for the `Game` class.

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

Include dependency graph for Game.cpp:



4.16.1 Detailed Description

Implementation file for the `Game` class.

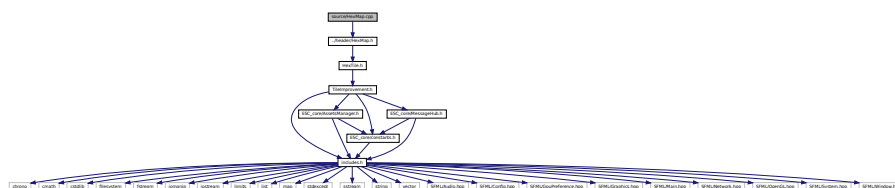
A class which defines a tile of a hex map.

4.17 source/HexMap.cpp File Reference

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

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

Include dependency graph for HexMap.cpp:



4.19.1 Detailed Description

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

4.19.2 Function Documentation

4.19.2.1 `constructRenderWindow()`

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

Helper function to construct render window.

Returns

Pointer to the render window.

```
88 {
89     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
90         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
91         "Road To Zero"
92     );
93
94     return render_window_ptr;
95 } /* constructRenderWindow() */
```

4.19.2.2 `loadAssets()`

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

Helper function to load game assets.

Parameters

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

```
32 {
33     // 1. load font assets
34     assets_manager_ptr->loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
35     assets_manager_ptr->loadFont("assets/fonts/Glass_TTY_VT220.ttf", "Glass_TTY_VT220");
36
37     // 2. load tile sheets
38     assets_manager_ptr->loadTexture(
39         "assets/tile_sheets/pine_tree_64x64_1.png",
40         "pine_tree_64x64_1"
41     );
42
43     assets_manager_ptr->loadTexture(
44         "assets/tile_sheets/wheat_64x64_1.png",
45         "wheat_64x64_1"
46     );
47
48     assets_manager_ptr->loadTexture(
49         "assets/tile_sheets/mountain_64x64_1.png",
50     );
```


Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL <https://www.sfml-dev.org/>. 118

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 117

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 72, 102

Index

- __assembleHexMap
 - HexMap, [51](#)
- __buildDrawOrderVector
 - HexMap, [51](#)
- __draw
 - Game, [38](#)
- __drawConsoleScreenFrame
 - ContextMenu, [20](#)
- __drawConsoleText
 - ContextMenu, [21](#)
- __drawFrameClockOverlay
 - Game, [38](#)
- __drawHUD
 - Game, [38](#)
- __drawVisualScreenFrame
 - ContextMenu, [21](#)
- __enforceOceanContinuity
 - HexMap, [52](#)
- __getMajorityTileType
 - HexMap, [53](#)
- __getNeighboursVector
 - HexMap, [54](#)
- __getNoise
 - HexMap, [54](#)
- __getSelectedTile
 - HexMap, [56](#)
- __getTileCoordsSubstring
 - HexTile, [73](#)
- __getTileImprovementSubstring
 - HexTile, [73](#)
- __getTileOptionsSubstring
 - HexTile, [74](#)
- __getTileResourceSubstring
 - HexTile, [74](#)
- __getTileTypeSubstring
 - HexTile, [75](#)
- __getValidMapIndexPositions
 - HexMap, [56](#)
- __handleKeyPressEvents
 - ContextMenu, [22](#)
 - Game, [39](#)
 - HexMap, [57](#)
 - HexTile, [76](#)
 - TileImprovement, [103](#)
- __handleMouseButtonEvents
 - ContextMenu, [22](#)
 - Game, [40](#)
 - HexMap, [57](#)
 - HexTile, [76](#)
- __insufficientCreditsAlarm
 - Game, [40](#)
- __isClicked
 - HexTile, [77](#)
- __isLakeTouchingOcean
 - HexMap, [58](#)
- __layTiles
 - HexMap, [58](#)
- __loadSoundBuffer
 - AssetsManager, [7](#)
- __procedurallyGenerateTileResources
 - HexMap, [61](#)
- __procedurallyGenerateTileTypes
 - HexMap, [61](#)
- __processEvent
 - Game, [41](#)
- __processMessage
 - Game, [42](#)
- __sendCreditsSpentMessage
 - HexTile, [77](#)
- __sendGameStateMessage
 - Game, [42](#)
- __sendGameStateRequest
 - HexTile, [78](#)
- __sendInsufficientCreditsMessage
 - HexTile, [78](#)
- __sendNoTileSelectedMessage
 - HexMap, [62](#)
- __sendQuitGameMessage
 - ContextMenu, [23](#)
- __sendRestartGameMessage
 - ContextMenu, [23](#)
- __sendTileSelectedMessage
 - HexTile, [78](#)
- __sendTileStateMessage
 - HexTile, [79](#)
- __setConsoleState
 - ContextMenu, [23](#)
- __setConsoleString
 - ContextMenu, [24](#)
- __setResourceText
 - HexTile, [79](#)
- __setUpConsoleScreen
 - ContextMenu, [24](#)
- __setUpConsoleScreenFrame
 - ContextMenu, [25](#)
- __setUpGlassScreen
 - HexMap, [62](#)

- __setUpMenuFrame
 - ContextMenu, 27
- __setUpNodeSprite
 - HexTile, 80
- __setUpResourceChipSprite
 - HexTile, 81
- __setUpSelectOutlineSprite
 - HexTile, 81
- __setUpTileSprite
 - HexTile, 81
- __setUpVisualScreen
 - ContextMenu, 27
- __setUpVisualScreenFrame
 - ContextMenu, 27
- __smoothTileTypes
 - HexMap, 62
- __toggleFrameClockOverlay
 - Game, 43
- ~AssetsManager
 - AssetsManager, 6
- ~ContextMenu
 - ContextMenu, 20
- ~Game
 - Game, 37
- ~HexMap
 - HexMap, 51
- ~HexTile
 - HexTile, 73
- ~MessageHub
 - MessageHub, 95
- ~TileImprovement
 - TileImprovement, 103
- ABOVE_AVERAGE
 - HexTile.h, 129
- addChannel
 - MessageHub, 95
- assess
 - HexMap, 63
 - HexTile, 82
- assets_manager_ptr
 - ContextMenu, 31
 - Game, 44
 - HexMap, 66
 - HexTile, 88
 - TileImprovement, 105
- AssetsManager, 5
 - __loadSoundBuffer, 7
 - ~AssetsManager, 6
 - AssetsManager, 6
 - clear, 8
 - current_track, 16
 - font_map, 16
 - getCurrentTrackKey, 9
 - getFont, 9
 - getSound, 10
 - getSoundBuffer, 10
 - getTexture, 11
 - getTrackStatus, 11
 - loadFont, 12
 - loadSound, 12
 - loadTexture, 13
 - loadTrack, 14
 - nextTrack, 14
 - pauseTrack, 15
 - playTrack, 15
 - previousTrack, 15
 - sound_map, 16
 - soundbuffer_map, 16
 - stopTrack, 15
 - texture_map, 16
 - track_map, 17
- AVERAGE
 - HexTile.h, 129
- BELOW_AVERAGE
 - HexTile.h, 129
- bool_payload
 - Message, 92
- border_tiles_vec
 - HexMap, 66
- channel
 - Message, 93
- clear
 - AssetsManager, 8
 - HexMap, 63
 - MessageHub, 96
- clearMessages
 - MessageHub, 96
- clock
 - Game, 44
- CO2E_KG_PER_LITRE_DIESEL
 - constants.h, 114
- console_screen
 - ContextMenu, 31
- console_screen_frame_bottom
 - ContextMenu, 31
- console_screen_frame_left
 - ContextMenu, 31
- console_screen_frame_right
 - ContextMenu, 31
- console_screen_frame_top
 - ContextMenu, 31
- console_state
 - ContextMenu, 32
- console_string
 - ContextMenu, 32
- ConsoleState
 - ContextMenu.h, 108
- constants.h
 - CO2E_KG_PER_LITRE_DIESEL, 114
 - EMISSIONS_LIFETIME_LIMIT_TONNES, 114
 - FLOAT_TOLERANCE, 114
 - FOREST_GREEN, 111
 - FRAMES_PER_SECOND, 114
 - GAME_CHANNEL, 114
 - GAME_HEIGHT, 115

- GAME_STATE_CHANNEL, 115
- GAME_WIDTH, 115
- LAKE_BLUE, 111
- MENU_FRAME_GREY, 112
- MONOCHROME_SCREEN_BACKGROUND, 112
- MONOCHROME_TEXT_AMBER, 112
- MONOCHROME_TEXT_GREEN, 112
- MONOCHROME_TEXT_RED, 112
- MOUNTAINS_GREY, 113
- NO_TILE_SELECTED_CHANNEL, 115
- OCEAN_BLUE, 113
- PLAINS_YELLOW, 113
- RESOURCE_ASSESSMENT_COST, 115
- RESOURCE_CHIP_GREY, 113
- SECONDS_PER_FRAME, 115
- SECONDS_PER_MONTH, 116
- SECONDS_PER_YEAR, 116
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES, 116
- TILE_SELECTED_CHANNEL, 116
- TILE_STATE_CHANNEL, 116
- TILE_TYPE_CUMULATIVE_PROBABILITIES, 116
- VISUAL_SCREEN_FRAME_GREY, 113
- constructRenderWindow
 - main.cpp, 141
- context_menu_ptr
 - Game, 44
- ContextMenu, 17
 - __drawConsoleScreenFrame, 20
 - __drawConsoleText, 21
 - __drawVisualScreenFrame, 21
 - __handleKeyPressEvents, 22
 - __handleMouseButtonEvents, 22
 - __sendQuitGameMessage, 23
 - __sendRestartGameMessage, 23
 - __setConsoleState, 23
 - __setConsoleString, 24
 - __setUpConsoleScreen, 24
 - __setUpConsoleScreenFrame, 25
 - __setUpMenuFrame, 27
 - __setUpVisualScreen, 27
 - __setUpVisualScreenFrame, 27
 - ~ContextMenu, 20
 - assets_manager_ptr, 31
 - console_screen, 31
 - console_screen_frame_bottom, 31
 - console_screen_frame_left, 31
 - console_screen_frame_right, 31
 - console_screen_frame_top, 31
 - console_state, 32
 - console_string, 32
 - ContextMenu, 19
 - draw, 29
 - event_ptr, 32
 - frame, 32
 - game_menu_up, 32
 - menu_frame, 32
 - message_hub_ptr, 33
 - position_x, 33
 - position_y, 33
 - processEvent, 29
 - processMessage, 30
 - render_window_ptr, 33
 - visual_screen, 33
 - visual_screen_frame_bottom, 33
 - visual_screen_frame_left, 34
 - visual_screen_frame_right, 34
 - visual_screen_frame_top, 34
- ContextMenu.h
 - ConsoleState, 108
 - MENU, 108
 - N_CONSOLE_STATES, 108
 - NONE_STATE, 108
 - READY, 108
 - TILE, 108
- credits
 - Game, 45
 - HexTile, 88
 - TileImprovement, 105
- cumulative_emissions_tonnes
 - Game, 45
- current_track
 - AssetsManager, 16
- decorateTile
 - HexTile, 82
- demand_MWh
 - Game, 45
- double_payload
 - Message, 93
- draw
 - ContextMenu, 29
 - HexMap, 64
 - HexTile, 83
 - TileImprovement, 104
- EMISSIONS_LIFETIME_LIMIT_TONNES
 - constants.h, 114
- ENERGY_STORAGE_SYSTEM
 - TileImprovement.h, 131
- event
 - Game, 45
- event_ptr
 - ContextMenu, 32
 - HexMap, 66
 - HexTile, 88
 - TileImprovement, 105
- expectedErrorNotDetected
 - testing_utils.cpp, 134
 - testing_utils.h, 120
- FLOAT_TOLERANCE
 - constants.h, 114
- font_map
 - AssetsManager, 16
- FOREST
 - HexTile.h, 129

FOREST_GREEN
 constants.h, 111
 frame
 ContextMenu, 32
 Game, 45
 HexMap, 66
 HexTile, 88
 TileImprovement, 105
 FRAMES_PER_SECOND
 constants.h, 114
 Game, 34
 __draw, 38
 __drawFrameClockOverlay, 38
 __drawHUD, 38
 __handleKeyPressEvents, 39
 __handleMouseButtonEvents, 40
 __insufficientCreditsAlarm, 40
 __processEvent, 41
 __processMessage, 42
 __sendGameStateMessage, 42
 __toggleFrameClockOverlay, 43
 ~Game, 37
 assets_manager_ptr, 44
 clock, 44
 context_menu_ptr, 44
 credits, 45
 cumulative_emissions_tonnes, 45
 demand_MWh, 45
 event, 45
 frame, 45
 Game, 36
 game_loop_broken, 45
 hex_map_ptr, 46
 message_hub, 46
 month, 46
 population, 46
 quit_game, 46
 render_window_ptr, 46
 run, 43
 show_frame_clock_overlay, 47
 time_since_start_s, 47
 year, 47
 GAME_CHANNEL
 constants.h, 114
 GAME_HEIGHT
 constants.h, 115
 game_loop_broken
 Game, 45
 game_menu_up
 ContextMenu, 32
 GAME_STATE_CHANNEL
 constants.h, 115
 GAME_WIDTH
 constants.h, 115
 getCurrentTrackKey
 AssetsManager, 9
 getFont
 AssetsManager, 9
 getSound
 AssetsManager, 10
 getSoundBuffer
 AssetsManager, 10
 getTexture
 AssetsManager, 11
 getTrackStatus
 AssetsManager, 11
 glass_screen
 HexMap, 67
 GOOD
 HexTile.h, 129
 has_improvement
 HexTile, 88
 hasTraffic
 MessageHub, 96
 header/ContextMenu.h, 107
 header/ESC_core/AssetsManager.h, 108
 header/ESC_core/constants.h, 109
 header/ESC_core/doxygen_cite.h, 117
 header/ESC_core/includes.h, 117
 header/ESC_core/MessageHub.h, 118
 header/ESC_core/testing_utils.h, 119
 header/Game.h, 126
 header/HexMap.h, 127
 header/HexTile.h, 128
 header/TileImprovement.h, 130
 hex_draw_order_vec
 HexMap, 67
 hex_map
 HexMap, 67
 hex_map_ptr
 Game, 46
 HexMap, 47
 __assembleHexMap, 51
 __buildDrawOrderVector, 51
 __enforceOceanContinuity, 52
 __getMajorityTileType, 53
 __getNeighboursVector, 54
 __getNoise, 54
 __getSelectedTile, 56
 __getValidMapIndexPositions, 56
 __handleKeyPressEvents, 57
 __handleMouseButtonEvents, 57
 __isLakeTouchingOcean, 58
 __layTiles, 58
 __procedurallyGenerateTileResources, 61
 __procedurallyGenerateTileTypes, 61
 __sendNoTileSelectedMessage, 62
 __setUpGlassScreen, 62
 __smoothTileTypes, 62
 ~HexMap, 51
 assess, 63
 assets_manager_ptr, 66
 border_tiles_vec, 66
 clear, 63
 draw, 64
 event_ptr, 66

- frame, 66
- glass_screen, 67
- hex_draw_order_vec, 67
- hex_map, 67
- HexMap, 50
- message_hub_ptr, 67
- n_layers, 67
- n_tiles, 67
- position_x, 68
- position_y, 68
- processEvent, 64
- processMessage, 65
- render_window_ptr, 68
- reroll, 65
- tile_position_x_vec, 68
- tile_position_y_vec, 68
- tile_selected, 68
- toggleResourceOverlay, 65
- HexTile, 69
 - __getTileCoordsSubstring, 73
 - __getTileImprovementSubstring, 73
 - __getTileOptionsSubstring, 74
 - __getTileResourceSubstring, 74
 - __getTileTypeSubstring, 75
 - __handleKeyPressEvents, 76
 - __handleMouseButtonEvents, 76
 - __isClicked, 77
 - __sendCreditsSpentMessage, 77
 - __sendGameStateRequest, 78
 - __sendInsufficientCreditsMessage, 78
 - __sendTileSelectedMessage, 78
 - __sendTileStateMessage, 79
 - __setResourceText, 79
 - __setUpNodeSprite, 80
 - __setUpResourceChipSprite, 81
 - __setUpSelectOutlineSprite, 81
 - __setUpTileSprite, 81
 - ~HexTile, 73
 - assess, 82
 - assets_manager_ptr, 88
 - credits, 88
 - decorateTile, 82
 - draw, 83
 - event_ptr, 88
 - frame, 88
 - has_improvement, 88
 - HexTile, 72
 - is_selected, 88
 - major_radius, 89
 - message_hub_ptr, 89
 - minor_radius, 89
 - node_sprite, 89
 - position_x, 89
 - position_y, 89
 - processEvent, 84
 - processMessage, 84
 - render_window_ptr, 90
 - resource_assessed, 90
 - resource_chip_sprite, 90
 - resource_text, 90
 - select_outline_sprite, 90
 - setTileResource, 85
 - setTileType, 86
 - settlement_built, 90
 - show_node, 91
 - show_resource, 91
 - tile_decoration_sprite, 91
 - tile_improvement_ptr, 91
 - tile_resource, 91
 - tile_sprite, 91
 - tile_type, 92
 - toggleResourceOverlay, 87
- HexTile.h
 - ABOVE_AVERAGE, 129
 - AVERAGE, 129
 - BELOW_AVERAGE, 129
 - FOREST, 129
 - GOOD, 129
 - LAKE, 129
 - MOUNTAINS, 129
 - N_TILE_RESOURCES, 129
 - N_TILE_TYPES, 129
 - NONE_TYPE, 129
 - OCEAN, 129
 - PLAINS, 129
 - POOR, 129
 - TileResource, 129
 - TileType, 129
- int_payload
 - Message, 93
- is_selected
 - HexTile, 88
- isEmpty
 - MessageHub, 96
- LAKE
 - HexTile.h, 129
- LAKE_BLUE
 - constants.h, 111
- loadAssets
 - main.cpp, 141
- loadFont
 - AssetsManager, 12
- loadSound
 - AssetsManager, 12
- loadTexture
 - AssetsManager, 13
- loadTrack
 - AssetsManager, 14
- main
 - main.cpp, 142
- main.cpp
 - constructRenderWindow, 141
 - loadAssets, 141
 - main, 142

- major_radius
 - HexTile, 89
- MENU
 - ContextMenu.h, 108
- menu_frame
 - ContextMenu, 32
- MENU_FRAME_GREY
 - constants.h, 112
- Message, 92
 - bool_payload, 92
 - channel, 93
 - double_payload, 93
 - int_payload, 93
 - string_payload, 93
 - subject, 93
- message_hub
 - Game, 46
- message_hub_ptr
 - ContextMenu, 33
 - HexMap, 67
 - HexTile, 89
 - TileImprovement, 105
- message_map
 - MessageHub, 100
- MessageHub, 94
 - ~MessageHub, 95
 - addChannel, 95
 - clear, 96
 - clearMessages, 96
 - hasTraffic, 96
 - isEmpty, 96
 - message_map, 100
 - MessageHub, 94
 - popMessage, 98
 - receiveMessage, 99
 - removeChannel, 99
 - sendMessage, 100
- minor_radius
 - HexTile, 89
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, 112
- MONOCHROME_TEXT_AMBER
 - constants.h, 112
- MONOCHROME_TEXT_GREEN
 - constants.h, 112
- MONOCHROME_TEXT_RED
 - constants.h, 112
- month
 - Game, 46
- MOUNTAINS
 - HexTile.h, 129
- MOUNTAINS_GREY
 - constants.h, 113
- N_CONSOLE_STATES
 - ContextMenu.h, 108
- n_layers
 - HexMap, 67
- N_TILE_IMPROVEMENT_TYPES
 - TileImprovement.h, 131
- N_TILE_RESOURCES
 - HexTile.h, 129
- N_TILE_TYPES
 - HexTile.h, 129
- n_tiles
 - HexMap, 67
- nextTrack
 - AssetsManager, 14
- NO_TILE_SELECTED_CHANNEL
 - constants.h, 115
- node_sprite
 - HexTile, 89
- NONE_STATE
 - ContextMenu.h, 108
- NONE_TYPE
 - HexTile.h, 129
- OCEAN
 - HexTile.h, 129
- OCEAN_BLUE
 - constants.h, 113
- pauseTrack
 - AssetsManager, 15
- PLAINS
 - HexTile.h, 129
- PLAINS_YELLOW
 - constants.h, 113
- playTrack
 - AssetsManager, 15
- POOR
 - HexTile.h, 129
- popMessage
 - MessageHub, 98
- population
 - Game, 46
- position_x
 - ContextMenu, 33
 - HexMap, 68
 - HexTile, 89
 - TileImprovement, 106
- position_y
 - ContextMenu, 33
 - HexMap, 68
 - HexTile, 89
 - TileImprovement, 106
- previousTrack
 - AssetsManager, 15
- printGold
 - testing_utils.cpp, 134
 - testing_utils.h, 121
- printGreen
 - testing_utils.cpp, 134
 - testing_utils.h, 121
- printRed
 - testing_utils.cpp, 135
 - testing_utils.h, 122
- processEvent

- ContextMenu, 29
- HexMap, 64
- HexTile, 84
- TileImprovement, 104
- processMessage
 - ContextMenu, 30
 - HexMap, 65
 - HexTile, 84
 - TileImprovement, 104
- quit_game
 - Game, 46
- READY
 - ContextMenu.h, 108
- receiveMessage
 - MessageHub, 99
- removeChannel
 - MessageHub, 99
- render_window_ptr
 - ContextMenu, 33
 - Game, 46
 - HexMap, 68
 - HexTile, 90
 - TileImprovement, 106
- reroll
 - HexMap, 65
- resource_assessed
 - HexTile, 90
- RESOURCE_ASSESSMENT_COST
 - constants.h, 115
- RESOURCE_CHIP_GREY
 - constants.h, 113
- resource_chip_sprite
 - HexTile, 90
- resource_text
 - HexTile, 90
- run
 - Game, 43
- SECONDS_PER_FRAME
 - constants.h, 115
- SECONDS_PER_MONTH
 - constants.h, 116
- SECONDS_PER_YEAR
 - constants.h, 116
- select_outline_sprite
 - HexTile, 90
- sendMessage
 - MessageHub, 100
- setTileResource
 - HexTile, 85
- setTileType
 - HexTile, 86
- SETTLEMENT
 - TileImprovement.h, 131
- settlement_built
 - HexTile, 90
- show_frame_clock_overlay
 - Game, 47
- show_node
 - HexTile, 91
- show_resource
 - HexTile, 91
- SOLAR_PV
 - TileImprovement.h, 131
- sound_map
 - AssetsManager, 16
- soundbuffer_map
 - AssetsManager, 16
- source/ContextMenu.cpp, 131
- source/ESC_core/AssetsManager.cpp, 132
- source/ESC_core/MessageHub.cpp, 132
- source/ESC_core/testing_utils.cpp, 133
- source/Game.cpp, 139
- source/HexMap.cpp, 139
- source/HexTile.cpp, 140
- source/main.cpp, 140
- source/TileImprovement.cpp, 142
- stopTrack
 - AssetsManager, 15
- string_payload
 - Message, 93
- subject
 - Message, 93
- testFloatEquals
 - testing_utils.cpp, 135
 - testing_utils.h, 122
- testGreaterThan
 - testing_utils.cpp, 136
 - testing_utils.h, 123
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 136
 - testing_utils.h, 123
- testing_utils.cpp
 - expectedErrorNotDetected, 134
 - printGold, 134
 - printGreen, 134
 - printRed, 135
 - testFloatEquals, 135
 - testGreaterThan, 136
 - testGreaterThanOrEqualTo, 136
 - testLessThan, 137
 - testLessThanOrEqualTo, 138
 - testTruth, 138
- testing_utils.h
 - expectedErrorNotDetected, 120
 - printGold, 121
 - printGreen, 121
 - printRed, 122
 - testFloatEquals, 122
 - testGreaterThan, 123
 - testGreaterThanOrEqualTo, 123
 - testLessThan, 124
 - testLessThanOrEqualTo, 125
 - testTruth, 125
- testLessThan

- testing_utils.cpp, 137
- testing_utils.h, 124
- testLessThanOrEqualTo
 - testing_utils.cpp, 138
 - testing_utils.h, 125
- testTruth
 - testing_utils.cpp, 138
 - testing_utils.h, 125
- texture_map
 - AssetsManager, 16
- TIDAL_TURBINE
 - TileImprovement.h, 131
- TILE
 - ContextMenu.h, 108
- tile_decoration_sprite
 - HexTile, 91
- tile_improvement_ptr
 - HexTile, 91
- tile_improvement_sprite_animated
 - TileImprovement, 106
- tile_improvement_sprite_static
 - TileImprovement, 106
- tile_position_x_vec
 - HexMap, 68
- tile_position_y_vec
 - HexMap, 68
- tile_resource
 - HexTile, 91
- TILE_RESOURCE_CUMULATIVE_PROBABILITIES
 - constants.h, 116
- tile_selected
 - HexMap, 68
- TILE_SELECTED_CHANNEL
 - constants.h, 116
- tile_sprite
 - HexTile, 91
- TILE_STATE_CHANNEL
 - constants.h, 116
- tile_type
 - HexTile, 92
- TILE_TYPE_CUMULATIVE_PROBABILITIES
 - constants.h, 116
- TileImprovement, 101
 - __handleKeyPressEvents, 103
 - __handleMouseButtonEvents, 103
 - ~TileImprovement, 103
 - assets_manager_ptr, 105
 - credits, 105
 - draw, 104
 - event_ptr, 105
 - frame, 105
 - message_hub_ptr, 105
 - position_x, 106
 - position_y, 106
 - processEvent, 104
 - processMessage, 104
 - render_window_ptr, 106
 - tile_improvement_sprite_animated, 106
 - tile_improvement_sprite_static, 106
 - TileImprovement, 102
- TileImprovement.h
 - ENERGY_STORAGE_SYSTEM, 131
 - N_TILE_IMPROVEMENT_TYPES, 131
 - SETTLEMENT, 131
 - SOLAR_PV, 131
 - TIDAL_TURBINE, 131
 - TileImprovementType, 131
 - WAVE_ENERGY_CONVERTER, 131
 - WIND_TURBINE, 131
- TileImprovementType
 - TileImprovement.h, 131
- TileResource
 - HexTile.h, 129
- TileType
 - HexTile.h, 129
- time_since_start_s
 - Game, 47
- toggleResourceOverlay
 - HexMap, 65
 - HexTile, 87
- track_map
 - AssetsManager, 17
- visual_screen
 - ContextMenu, 33
- visual_screen_frame_bottom
 - ContextMenu, 33
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, 113
- visual_screen_frame_left
 - ContextMenu, 34
- visual_screen_frame_right
 - ContextMenu, 34
- visual_screen_frame_top
 - ContextMenu, 34
- WAVE_ENERGY_CONVERTER
 - TileImprovement.h, 131
- WIND_TURBINE
 - TileImprovement.h, 131
- year
 - Game, 47