

HelloWorld

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()	20
3.2.3.3	__drawVisualScreenFrame()	21
3.2.3.4	__setUpConsoleScreen()	21
3.2.3.5	__setUpConsoleScreenFrame()	22
3.2.3.6	__setUpMenuFrame()	24
3.2.3.7	__setUpVisualScreen()	24
3.2.3.8	__setUpVisualScreenFrame()	24
3.2.3.9	draw()	26
3.2.3.10	process()	26
3.2.4	Member Data Documentation	27
3.2.4.1	assets_manager_ptr	27
3.2.4.2	console_message	27
3.2.4.3	console_screen	27
3.2.4.4	console_screen_frame_bottom	28
3.2.4.5	console_screen_frame_left	28
3.2.4.6	console_screen_frame_right	28
3.2.4.7	console_screen_frame_top	28
3.2.4.8	frame	28
3.2.4.9	game_menu_up	28
3.2.4.10	inputs_handler_ptr	29
3.2.4.11	menu_frame	29
3.2.4.12	messages_handler_ptr	29
3.2.4.13	position_x	29
3.2.4.14	position_y	29
3.2.4.15	render_window_ptr	29
3.2.4.16	visual_screen	30
3.2.4.17	visual_screen_frame_bottom	30
3.2.4.18	visual_screen_frame_left	30
3.2.4.19	visual_screen_frame_right	30
3.2.4.20	visual_screen_frame_top	30
3.3	HexMap Class Reference	31
3.3.1	Detailed Description	33
3.3.2	Constructor & Destructor Documentation	33
3.3.2.1	HexMap()	33
3.3.2.2	~HexMap()	33
3.3.3	Member Function Documentation	34
3.3.3.1	__assembleHexMap()	34
3.3.3.2	__enforceOceanContinuity()	34
3.3.3.3	__getMajorityTileType()	35
3.3.3.4	__getNeighboursVector()	36
3.3.3.5	__getNoise()	37

3.3.3.6	__getSelectedTile()	38
3.3.3.7	__getValidMapIndexPositions()	39
3.3.3.8	__isLakeTouchingOcean()	39
3.3.3.9	__layTiles()	40
3.3.3.10	__procedurallyGenerateTileResources()	42
3.3.3.11	__procedurallyGenerateTileTypes()	42
3.3.3.12	__setUpGlassScreen()	43
3.3.3.13	__smoothTileTypes()	43
3.3.3.14	assess()	44
3.3.3.15	clear()	44
3.3.3.16	draw()	45
3.3.3.17	process()	45
3.3.3.18	reroll()	46
3.3.3.19	toggleResourceOverlay()	46
3.3.4	Member Data Documentation	46
3.3.4.1	assets_manager_ptr	46
3.3.4.2	border_tiles_vec	46
3.3.4.3	frame	47
3.3.4.4	glass_screen	47
3.3.4.5	hex_map	47
3.3.4.6	inputs_handler_ptr	47
3.3.4.7	messages_handler_ptr	47
3.3.4.8	n_layers	47
3.3.4.9	n_tiles	48
3.3.4.10	position_x	48
3.3.4.11	position_y	48
3.3.4.12	render_window_ptr	48
3.3.4.13	tile_position_x_vec	48
3.3.4.14	tile_position_y_vec	48
3.4	HexTile Class Reference	49
3.4.1	Detailed Description	51
3.4.2	Constructor & Destructor Documentation	51
3.4.2.1	HexTile()	51
3.4.2.2	~HexTile()	52
3.4.3	Member Function Documentation	52
3.4.3.1	__isClicked()	52
3.4.3.2	__setResourceText()	53
3.4.3.3	__setUpNodeSprite()	53
3.4.3.4	__setUpResourceChipSprite()	54
3.4.3.5	__setUpSelectOutlineSprite()	54
3.4.3.6	__setUpTileSprite()	55
3.4.3.7	assess()	55

3.4.3.8 draw()	55
3.4.3.9 process()	56
3.4.3.10 setTileResource() [1/2]	56
3.4.3.11 setTileResource() [2/2]	57
3.4.3.12 setTileType() [1/2]	57
3.4.3.13 setTileType() [2/2]	58
3.4.3.14 toggleResourceOverlay()	59
3.4.4 Member Data Documentation	59
3.4.4.1 assets_manager_ptr	59
3.4.4.2 frame	59
3.4.4.3 inputs_handler_ptr	59
3.4.4.4 is_selected	60
3.4.4.5 major_radius	60
3.4.4.6 messages_handler_ptr	60
3.4.4.7 minor_radius	60
3.4.4.8 node_sprite	60
3.4.4.9 position_x	60
3.4.4.10 position_y	61
3.4.4.11 render_window_ptr	61
3.4.4.12 resource_assessed	61
3.4.4.13 resource_chip_sprite	61
3.4.4.14 resource_text	61
3.4.4.15 select_outline_sprite	61
3.4.4.16 show_node	62
3.4.4.17 show_resource	62
3.4.4.18 tile_resource	62
3.4.4.19 tile_sprite	62
3.4.4.20 tile_type	62
3.5 InputsHandler Class Reference	62
3.5.1 Detailed Description	63
3.5.2 Constructor & Destructor Documentation	63
3.5.2.1 InputsHandler()	63
3.5.2.2 ~InputsHandler()	64
3.5.3 Member Function Documentation	64
3.5.3.1 __constructKeyCodeMap()	64
3.5.3.2 printKeysPressed()	68
3.5.3.3 process()	68
3.5.3.4 reset()	69
3.5.4 Member Data Documentation	69
3.5.4.1 key_code_map	69
3.5.4.2 key_press_vec	69
3.5.4.3 key_pressed_once_vec	70

3.5.4.4 mouse_left_click	70
3.5.4.5 mouse_right_click	70
3.6 MessagesHandler Class Reference	70
3.6.1 Detailed Description	70
3.6.2 Constructor & Destructor Documentation	71
3.6.2.1 MessagesHandler()	71
3.6.2.2 ~MessagesHandler()	71
3.6.3 Member Function Documentation	71
3.6.3.1 process()	71
4 File Documentation	73
4.1 header/ContextMenu/ContextMenu.h File Reference	73
4.1.1 Detailed Description	74
4.2 header/ESC_core/AssetsManager.h File Reference	74
4.2.1 Detailed Description	74
4.3 header/ESC_core/constants.h File Reference	75
4.3.1 Detailed Description	76
4.3.2 Function Documentation	76
4.3.2.1 FOREST_GREEN()	76
4.3.2.2 LAKE_BLUE()	76
4.3.2.3 MENU_FRAME_GREY()	77
4.3.2.4 MONOCHROME_SCREEN_BACKGROUND()	77
4.3.2.5 MONOCHROME_TEXT_AMBER()	77
4.3.2.6 MONOCHROME_TEXT_GREEN()	77
4.3.2.7 MONOCHROME_TEXT_RED()	77
4.3.2.8 MOUNTAINS_GREY()	78
4.3.2.9 OCEAN_BLUE()	78
4.3.2.10 PLAINS_YELLOW()	78
4.3.2.11 VISUAL_SCREEN_FRAME_GREY()	78
4.3.3 Variable Documentation	78
4.3.3.1 FLOAT_TOLERANCE	79
4.3.3.2 FRAMES_PER_SECOND	79
4.3.3.3 GAME_HEIGHT	79
4.3.3.4 GAME_WIDTH	79
4.3.3.5 SECONDS_PER_FRAME	79
4.3.3.6 tile_resource_cumulative_probabilities	79
4.3.3.7 tile_type_cumulative_probabilities	80
4.4 header/ESC_core/doxygen_cite.h File Reference	80
4.4.1 Detailed Description	80
4.5 header/ESC_core/includes.h File Reference	80
4.5.1 Detailed Description	81
4.6 header/ESC_core/InputsHandler.h File Reference	81

4.6.1 Detailed Description	82
4.7 header/ESC_core/MessagesHandler.h File Reference	82
4.7.1 Detailed Description	82
4.8 header/ESC_core/testing_utils.h File Reference	83
4.8.1 Detailed Description	83
4.8.2 Function Documentation	84
4.8.2.1 expectedErrorNotDetected()	84
4.8.2.2 printGold()	84
4.8.2.3 printGreen()	84
4.8.2.4 printRed()	85
4.8.2.5 testFloatEquals()	85
4.8.2.6 testGreaterThan()	86
4.8.2.7 testGreaterThanOrEqualTo()	86
4.8.2.8 testLessThan()	87
4.8.2.9 testLessThanOrEqualTo()	88
4.8.2.10 testTruth()	88
4.9 header/HexMap/HexMap.h File Reference	89
4.9.1 Detailed Description	89
4.10 header/HexMap/HexTile.h File Reference	90
4.10.1 Detailed Description	90
4.10.2 Enumeration Type Documentation	91
4.10.2.1 TileResource	91
4.10.2.2 TileType	91
4.11 source/ContextMenu/ContextMenu.cpp File Reference	92
4.11.1 Detailed Description	92
4.12 source/ESC_core/AssetsManager.cpp File Reference	92
4.12.1 Detailed Description	92
4.13 source/ESC_core/InputsHandler.cpp File Reference	92
4.13.1 Detailed Description	93
4.14 source/ESC_core/MessagesHandler.cpp File Reference	93
4.14.1 Detailed Description	93
4.15 source/ESC_core/testing_utils.cpp File Reference	93
4.15.1 Detailed Description	94
4.15.2 Function Documentation	94
4.15.2.1 expectedErrorNotDetected()	94
4.15.2.2 printGold()	95
4.15.2.3 printGreen()	95
4.15.2.4 printRed()	95
4.15.2.5 testFloatEquals()	96
4.15.2.6 testGreaterThan()	96
4.15.2.7 testGreaterThanOrEqualTo()	97
4.15.2.8 testLessThan()	98

4.15.2.9 testLessThanOrEqualTo()	98
4.15.2.10 testTruth()	99
4.16 source/HexMap/HexMap.cpp File Reference	99
4.16.1 Detailed Description	100
4.17 source/HexMap/HexTile.cpp File Reference	100
4.17.1 Detailed Description	100
4.18 test/ContextMenu/test_ContextMenu.cpp File Reference	100
4.18.1 Detailed Description	101
4.18.2 Function Documentation	101
4.18.2.1 main()	101
4.19 test/ESC_core/test_AssetsManager.cpp File Reference	103
4.19.1 Detailed Description	103
4.19.2 Function Documentation	103
4.19.2.1 main()	104
4.20 test/ESC_core/test_InputsHandler.cpp File Reference	106
4.20.1 Detailed Description	106
4.20.2 Function Documentation	106
4.20.2.1 main()	107
4.21 test/ESC_core/test_MessagesHandler.cpp File Reference	108
4.21.1 Detailed Description	108
4.21.2 Function Documentation	109
4.21.2.1 main()	109
4.22 test/HexMap/test_HexMap.cpp File Reference	110
4.22.1 Detailed Description	110
4.22.2 Function Documentation	110
4.22.2.1 main()	111
Bibliography	113
Index	115

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
HexMap	A class which defines a hex map of hex tiles	31
HexTile	A class which defines a hex tile of the hex map	49
InputsHandler	A class which handles inputs from peripherals (i.e., keyboard and mouse)	62
MessagesHandler	A class which handles message traffic between game objects	70

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

header/ContextMenu/ ContextMenu.h	
Header file for the ContextMenu class	73
header/ESC_core/ AssetsManager.h	
Header file for the AssetsManager class	74
header/ESC_core/ constants.h	
Header file for various constants	75
header/ESC_core/ doxygen_cite.h	
Header file which simply cites the doxygen tool	80
header/ESC_core/ includes.h	
Header file for various includes	80
header/ESC_core/ InputsHandler.h	
Header file for the InputsHandler class	81
header/ESC_core/ MessagesHandler.h	
Header file for the MessagesHandler class	82
header/ESC_core/ testing_utils.h	
Header file for various testing utilities	83
header/HexMap/ HexMap.h	
Header file for the HexMap class	89
header/HexMap/ HexTile.h	
Header file for the HexTile class	90
source/ContextMenu/ ContextMenu.cpp	
Implementation file for the ContextMenu class	92
source/ESC_core/ AssetsManager.cpp	
Implementation file for the AssetsManager class	92
source/ESC_core/ InputsHandler.cpp	
Implementation file for the InputsHandler class	92
source/ESC_core/ MessagesHandler.cpp	
Implementation file for the MessagesHandler class	93
source/ESC_core/ testing_utils.cpp	
Implementation file for various testing utilities	93
source/HexMap/ HexMap.cpp	
Implementation file for the HexMap class	99
source/HexMap/ HexTile.cpp	
Implementation file for the HexTile class	100
test/ContextMenu/ test_ContextMenu.cpp	
Suite of tests for the ContextMenu class	100

test/ESC_core/ test_AssetsManager.cpp	
Suite of tests for the AssetsManager class	103
test/ESC_core/ test_InputsHandler.cpp	
Suite of tests for the InputsHandler class	106
test/ESC_core/ test_MessagesHandler.cpp	
Suite of tests for the MessagesHandler class	108
test/HexMap/ test_HexMap.cpp	
Suite of tests for the HexMap class	110

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 }
705
706

```

```

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:

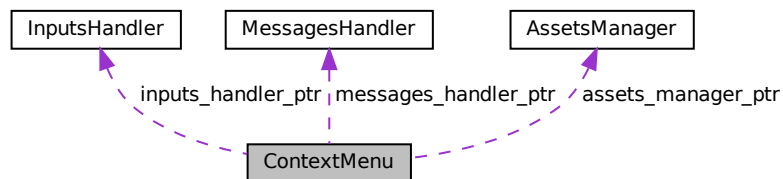
- header/ESC_core/[AssetsManager.h](#)
- source/ESC_core/[AssetsManager.cpp](#)

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](#) ([AssetsManager](#) *, [InputsHandler](#) *, [MessagesHandler](#) *, sf::RenderWindow *)
Constructor for the [ContextMenu](#) class.
- void [process](#) (void)
Method to process [ContextMenu](#). To be called once per frame.
- 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

- 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_message](#)
The message 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 [__drawConsoleText](#) (void)
Helper method to draw animated text to context menu console screen.

Private Attributes

- [AssetsManager](#) * [assets_manager_ptr](#)
A pointer to the assets manager.
- [InputsHandler](#) * [inputs_handler_ptr](#)
A pointer to the inputs handler.
- [MessagesHandler](#) * [messages_handler_ptr](#)
A pointer to the messages handler.
- [sf::RenderWindow](#) * [render_window_ptr](#)
A pointer to the render window.

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 (
    AssetsManager * assets_manager_ptr,
    InputsHandler * inputs_handler_ptr,
    MessagesHandler * messages_handler_ptr,
    sf::RenderWindow * render_window_ptr )
```

Constructor for the [ContextMenu](#) class.

Parameters

<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>inputs_handler_ptr</i>	Pointer to the inputs handler.
<i>messages_handler_ptr</i>	Pointer to the messages handler.
<i>render_window_ptr</i>	Pointer to the render window.

```
553 {
554     // 1. set attributes
555     this->assets_manager_ptr = assets_manager_ptr;
556     this->inputs_handler_ptr = inputs_handler_ptr;
557     this->messages_handler_ptr = messages_handler_ptr;
558     this->render_window_ptr = render_window_ptr;
559
560     this->game_menu_up = false;
561
562     this->frame = 0;
563
564     this->position_x = GAME_WIDTH;
565     this->position_y = 0;
566
567     this->console_message = "";
568
569     // 2. set up and position drawable attributes
570     this->__setUpMenuFrame();
571     this->__setUpVisualScreen();
572     this->__setUpVisualScreenFrame();
573     this->__setUpConsoleScreen();
```

```

574     this->__setUpConsoleScreenFrame();
575
576     std::cout << "ContextMenu constructed at " << this << std::endl;
577
578     return;
579 } /* ContextMenu() */

```

3.2.2.2 ~ContextMenu()

```

ContextMenu::~~ContextMenu (
    void )

```

Destructor for the [ContextMenu](#) class.

```

692 {
693     std::cout << "ContextMenu at " << this << " destroyed" << std::endl;
694
695     return;
696 } /* ~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.

```

455 {
456     // 1. init console text
457     sf::Text console_text;
458
459     if (this->console_message.empty()) {
460
461         // 32 char x 16 line console "-----\n";
462         std::string console_string = " **** RTZ 64 CONTEXT V12 **** \n";
463         console_string += " \n";
464         console_string += "64K RAM SYSTEM 38911 BYTES FREE\n";
465         console_string += " \n";
466         console_string += "[ESC]: MENU \n";
467         console_string += "[LEFT CLICK TILE]: TILE OPTIONS \n";
468         console_string += " \n";

```

```

469         console_string += "READY";
470
471         console_text.setString(console_string);
472     }
473
474     else {
475
476         // 32 char x 16 line console "-----\n";
477         std::string console_string = this->console_message;
478         console_string += "\nFRAME: ";
479         console_string += std::to_string(this->frame);
480
481         console_text.setString(console_string);
482     }
483
484
485     // 2. set console text font, size, colour, and position
486     console_text.setFont(*(assets_manager_ptr->getFont("Glass_TTY_VT220")));
487     console_text.setCharacterSize(16);
488
489     console_text.setFillColor(MONOCROME_TEXT_GREEN);
490
491     console_text.setPosition(
492         this->position_x - 50 - 300 + 16,
493         this->position_y + GAME_HEIGHT - 50 - 340 + 16
494     );
495
496
497     // 3. draw console text
498     this->render_window_ptr->draw(console_text);
499
500
501     // 4. assemble and draw blinking console cursor
502     if ((this->frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
503         sf::RectangleShape console_cursor(sf::Vector2f(10, 16));
504
505         console_cursor.setFillColor(MONOCROME_TEXT_GREEN);
506
507         console_cursor.setPosition(
508             console_text.getPosition().x,
509             console_text.getPosition().y + console_text.getLocalBounds().height + 10
510         );
511
512         this->render_window_ptr->draw(console_cursor);
513     }
514
515     return;
516 } /* __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 __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() */

```

3.2.3.5 __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.setFillColor(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.setFillColor(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.setFillColors(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() */

```

3.2.3.6 __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.setFillColors(MENU_FRAME_GREY);
39
40     return;
41 } /* __setUpMenuFrame() */

```

3.2.3.7 __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.setFillColors(MONOCHROME_SCREEN_BACKGROUND);
61
62     return;
63 } /* __setUpVisualScreen() */

```

3.2.3.8 __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.setFillColors(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     // 2. left framing
109     this->visual_screen_frame_left.setPointCount(n_points);
110
111     this->visual_screen_frame_left.setPoint(
112         0,
113         sf::Vector2f(this->position_x - 350, this->position_y + 50)
114     );
115     this->visual_screen_frame_left.setPoint(
116         1,
117         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
118     );
119     this->visual_screen_frame_left.setPoint(
120         2,
121         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
122     );
123     this->visual_screen_frame_left.setPoint(
124         3,
125         sf::Vector2f(this->position_x - 350, this->position_y + 350)
126     );
127
128     this->visual_screen_frame_left.setFillColors(VISUAL_SCREEN_FRAME_GREY);
129
130     this->visual_screen_frame_left.setOutlineThickness(2);
131     this->visual_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
132
133     this->visual_screen_frame_left.move(-2, 0);
134
135     // 3. bottom framing
136     this->visual_screen_frame_bottom.setPointCount(n_points);
137
138     this->visual_screen_frame_bottom.setPoint(
139         0,
140         sf::Vector2f(this->position_x - 350, this->position_y + 350)
141     );
142     this->visual_screen_frame_bottom.setPoint(
143         1,
144         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
145     );
146     this->visual_screen_frame_bottom.setPoint(
147         2,
148         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
149     );
150     this->visual_screen_frame_bottom.setPoint(
151         3,
152         sf::Vector2f(this->position_x - 50, this->position_y + 350)
153     );
154
155     this->visual_screen_frame_bottom.setFillColors(VISUAL_SCREEN_FRAME_GREY);
156
157     this->visual_screen_frame_bottom.setOutlineThickness(2);
158     this->visual_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
159
160     this->visual_screen_frame_bottom.move(0, 2);
161
162     // 4. right framing
163     this->visual_screen_frame_right.setPointCount(n_points);
164
165     this->visual_screen_frame_right.setPoint(
166         0,

```

```

170         sf::Vector2f(this->position_x - 50, this->position_y + 350)
171     );
172     this->visual_screen_frame_right.setPoint(
173         1,
174         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
175     );
176     this->visual_screen_frame_right.setPoint(
177         2,
178         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
179     );
180     this->visual_screen_frame_right.setPoint(
181         3,
182         sf::Vector2f(this->position_x - 50, this->position_y + 50)
183     );
184
185     this->visual_screen_frame_right.setFill_color(VISUAL_SCREEN_FRAME_GREY);
186
187     this->visual_screen_frame_right.setOutlineThickness(2);
188     this->visual_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
189
190     this->visual_screen_frame_right.move(2, 0);
191
192     return;
193 } /* __setUpVisualScreenFrame() */

```

3.2.3.9 draw()

```

void ContextMenu::draw (
    void )

```

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

```

662 {
663     // 1. menu frame
664     this->render_window_ptr->draw(this->menu_frame);
665
666     // 2. visual screen
667     this->render_window_ptr->draw(this->visual_screen);
668     this->__drawVisualScreenFrame();
669
670     // 3. console screen
671     this->render_window_ptr->draw(this->console_screen);
672     this->__drawConsoleScreenFrame();
673     this->__drawConsoleText();
674
675     this->frame++;
676     return;
677 } /* draw() */

```

3.2.3.10 process()

```

void ContextMenu::process (
    void )

```

Method to process `ContextMenu`. To be called once per frame.

```

602 {
603     // 1. handle inputs
604     if (this->inputs_handler_ptr->key_pressed_once_vec[sf::Keyboard::Escape]) {
605         if (not this->game_menu_up) {
606             this->game_menu_up = true;
607
608             // 32 char x 16 line console "-----\n";
609             std::string game_menu_string = "          **** MENU ****\n";
610             game_menu_string += "\n";
611             game_menu_string += "[T]:  TUTORIAL\n";
612             game_menu_string += "\n";
613             game_menu_string += "[R]:  RESTART\n";
614             game_menu_string += "\n";
615             game_menu_string += "\n";
616             game_menu_string += "\n";

```

```

617         game_menu_string += " \n";
618         game_menu_string += " \n";
619         game_menu_string += " \n";
620         game_menu_string += " \n";
621         game_menu_string += "[Q]:    QUIT \n";
622         game_menu_string += " \n";
623         game_menu_string += "[ESC]:  CLOSE MENU \n";
624
625         this->console_message = game_menu_string;
626     }
627
628     else {
629         this->game_menu_up = false;
630         this->console_message.clear();
631     }
632 }
633
634 if (this->inputs_handler_ptr->key_pressed_once_vec[sf::Keyboard::Q]) {
635     if (this->game_menu_up) {
636         this->render_window_ptr->close();
637     }
638 }
639
640 if (this->inputs_handler_ptr->mouse_right_click) {
641     this->game_menu_up = false;
642     this->console_message.clear();
643 }
644
645 return;
646 } /* process() */

```

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_message

`std::string` ContextMenu::console_message

The message to be printed to the console screen.

3.2.4.3 console_screen

`sf::RectangleShape` ContextMenu::console_screen

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

3.2.4.4 console_screen_frame_bottom

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

The bottom framing of the console screen.

3.2.4.5 console_screen_frame_left

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

The left framing of the console screen.

3.2.4.6 console_screen_frame_right

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

The right framing of the console screen.

3.2.4.7 console_screen_frame_top

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

The top framing of the console screen.

3.2.4.8 frame

```
int ContextMenu::frame
```

The current frame of this object.

3.2.4.9 game_menu_up

```
bool ContextMenu::game_menu_up
```

Indicates whether or not the game menu is up.

3.2.4.10 inputs_handler_ptr

```
InputsHandler* ContextMenu::inputs_handler_ptr [private]
```

A pointer to the inputs handler.

3.2.4.11 menu_frame

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

The frame of the context menu.

3.2.4.12 messages_handler_ptr

```
MessagesHandler* ContextMenu::messages_handler_ptr [private]
```

A pointer to the messages handler.

3.2.4.13 position_x

```
double ContextMenu::position_x
```

The position of the object.

3.2.4.14 position_y

```
double ContextMenu::position_y
```

The position of the object.

3.2.4.15 render_window_ptr

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

A pointer to the render window.

3.2.4.16 visual_screen

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

The context menu screen for visuals.

3.2.4.17 visual_screen_frame_bottom

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

The bottom framing of the visual screen.

3.2.4.18 visual_screen_frame_left

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

The left framing of the visual screen.

3.2.4.19 visual_screen_frame_right

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

The right framing of the visual screen.

3.2.4.20 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:

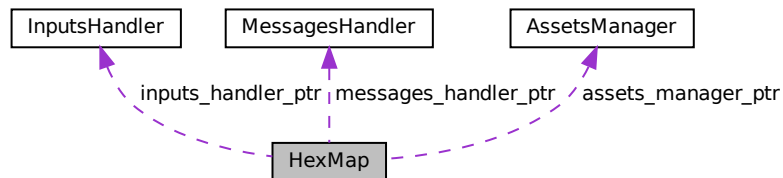
- header/ContextMenu/[ContextMenu.h](#)
- source/ContextMenu/[ContextMenu.cpp](#)

3.3 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, [AssetsManager](#) *, [InputsHandler](#) *, [MessagesHandler](#) *, sf::RenderWindow *)
Constructor for the [HexMap](#) class.
- void [assess](#) (void)
Method to assess the resource of the selected tile.
- void [process](#) (void)
Method to process [HexMap](#). To be called once per frame.
- void [reroll](#) (void)
Method to re-roll the hex map.
- void [toggleResourceOverlay](#) (void)
Method to toggle the hex map resource overlay.
- 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

- 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.

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.
- `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.

Private Attributes

- `AssetsManager *` [assets_manager_ptr](#)

A pointer to the assets manager.
- `InputsHandler *` [inputs_handler_ptr](#)

A pointer to the inputs handler.
- `MessagesHandler *` [messages_handler_ptr](#)

A pointer to the messages handler.
- `sf::RenderWindow *` [render_window_ptr](#)

A pointer to the render window.

3.3.1 Detailed Description

A class which defines a hex map of hex tiles.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 HexMap()

```
HexMap::HexMap (
    int n_layers,
    AssetsManager * assets_manager_ptr,
    InputsHandler * inputs_handler_ptr,
    MessagesHandler * messages_handler_ptr,
    sf::RenderWindow * render_window_ptr )
```

Constructor for the [HexMap](#) class.

Parameters

<i>n_layers</i>	The number of layers in the HexMap .
<i>assets_manager_ptr</i>	Pointer to the assets manager.
<i>inputs_handler_ptr</i>	Pointer to the inputs handler.
<i>messages_handler_ptr</i>	Pointer to the messages handler.
<i>render_window_ptr</i>	Pointer to the render window.

```
867 {
868     // 1. set attributes
869     this->assets_manager_ptr = assets_manager_ptr;
870     this->inputs_handler_ptr = inputs_handler_ptr;
871     this->messages_handler_ptr = messages_handler_ptr;
872     this->render_window_ptr = render_window_ptr;
873
874     this->frame = 0;
875
876     this->n_layers = n_layers;
877     if (this->n_layers < 0) {
878         this->n_layers = 0;
879     }
880
881     this->position_x = 400;
882     this->position_y = 400;
883
884     // 2. assemble n layer hex map
885     this->__assembleHexMap();
886
887     // 3. set up and position drawable attributes
888     this->__setUpGlassScreen();
889
890     std::cout << "HexMap constructed at " << this << std::endl;
891
892     return;
893 } /* HexMap() */
```

3.3.2.2 ~HexMap()

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

Destructor for the [HexMap](#) class.

```
1102 {
1103     this->clear();
1104
1105     std::cout << "HexMap at " << this << " destroyed" << std::endl;
1106
1107     return;
1108 } /* ~HexMap() */
```

3.3.3 Member Function Documentation

3.3.3.1 __assembleHexMap()

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

Helper method to assemble the hex map.

```
758 {
759     // 1. seed RNG (using milliseconds since 1 Jan 1970)
760     unsigned long long int milliseconds_since_epoch =
761         std::chrono::duration_cast<std::chrono::milliseconds>(
762             std::chrono::system_clock::now().time_since_epoch()
763         ).count();
764     srand(milliseconds_since_epoch);
765
766     // 2. lay tiles
767     this->__layTiles();
768
769     // 3. procedurally generate types
770     this->__procedurallyGenerateTileTypes();
771
772     // 4. procedurally generate resources
773     this->__procedurallyGenerateTileResources();
774
775     return;
776 } /* __assembleHexMap() */
```

3.3.3.2 __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.

```
669 {
670     std::cout << "enforcing ocean continuity ..." << std::endl;
671
672     bool tile_changed = false;
673
674     // 1. scan tiles and enforce (where appropriate)
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     for (
679         hex_map_iter_x = this->hex_map.begin();
680         hex_map_iter_x != this->hex_map.end();
681         hex_map_iter_x++
682     ) {
683         for (
684             hex_map_iter_y = hex_map_iter_x->second.begin();
685             hex_map_iter_y != hex_map_iter_x->second.end();
686             hex_map_iter_y++
687         ) {
688             hex_ptr = hex_map_iter_y->second;
689         }
690     }
```

```

690         if (this->__isLakeTouchingOcean(hex_ptr)) {
691             hex_ptr->setTileType(TileType :: OCEAN);
692             tile_changed = true;
693         }
694     }
695 }
696
697 if (tile_changed) {
698     this->__enforceOceanContinuity();
699 }
700 else {
701     return;
702 }
703 } /* __enforceOceanContinuity() */

```

3.3.3.3 __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.

```

525 {
526     // 1. init type count map
527     std::map<TileType, int> type_count_map;
528     type_count_map[hex_ptr->tile_type] = 1;
529
530     // 2. survey neighbours, count type instances
531     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
532
533     for (size_t i = 0; i < neighbours_vec.size(); i++) {
534         if (type_count_map.count(neighbours_vec[i]->tile_type) <= 0) {
535             type_count_map[neighbours_vec[i]->tile_type] = 1;
536         }
537         else {
538             type_count_map[neighbours_vec[i]->tile_type] += 1;
539         }
540     }
541
542     // 3. find majority tile type
543     int max_count = -1 * std::numeric_limits<int>::infinity();
544     TileType majority_tile_type = hex_ptr->tile_type;
545
546     std::map<TileType, int>::iterator map_iter;
547     for (
548         map_iter = type_count_map.begin();
549         map_iter != type_count_map.end();
550         map_iter++)
551     ){
552         if (map_iter->second > max_count) {
553             max_count = map_iter->second;
554             majority_tile_type = map_iter->first;
555         }
556     }
557
558     // 4. detect ties
559     for (
560         map_iter = type_count_map.begin();
561         map_iter != type_count_map.end();
562         map_iter++)
563     ){

```

```

564         if (
565             map_iter->second == max_count and
566             map_iter->first != majority_tile_type
567         ) {
568             majority_tile_type = hex_ptr->tile_type;
569             break;
570         }
571     }
572
573     return majority_tile_type;
574 } /* __getMajorityTileType() */

```

3.3.3.4 __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.

```

467 {
468     std::vector<HexTile*> neighbours_vec;
469
470     // 1. build potential neighbour positions
471     std::vector<double> potential_neighbour_x_vec(6, 0);
472     std::vector<double> potential_neighbour_y_vec(6, 0);
473
474     for (int i = 0; i < 6; i++) {
475         potential_neighbour_x_vec[i] = hex_ptr->position_x +
476             2 * hex_ptr->minor_radius * cos((60 * i) * (M_PI / 180));
477
478         potential_neighbour_y_vec[i] = hex_ptr->position_y +
479             2 * hex_ptr->minor_radius * sin((60 * i) * (M_PI / 180));
480     }
481
482     // 2. populate neighbours vector
483     std::vector<double> map_index_positions;
484     double potential_x = 0;
485     double potential_y = 0;
486
487     for (int i = 0; i < 6; i++) {
488         potential_x = potential_neighbour_x_vec[i];
489         potential_y = potential_neighbour_y_vec[i];
490
491         map_index_positions = this->__getValidMapIndexPositions(
492             potential_x,
493             potential_y
494         );
495
496         if (not (map_index_positions[0] == -1)) {
497             neighbours_vec.push_back(
498                 this->hex_map[map_index_positions[0]][map_index_positions[1]]
499             );
500         }
501     }
502
503     return neighbours_vec;
504 } /* __getNeighbourVector() */

```

3.3.3.5 __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].

```
247 {
248     // 1. generate random amplitude, wave number, direction, and phase vectors
249     std::vector<double> random_amplitude_vec(n_components, 0);
250     std::vector<double> random_wave_number_vec(n_components, 0);
251     std::vector<double> random_frequency_vec(n_components, 0);
252     std::vector<double> random_direction_vec(n_components, 0);
253     std::vector<double> random_phase_vec(n_components, 0);
254
255     for (int i = 0; i < n_components; i++) {
256         random_amplitude_vec[i] = 10 * ((double)rand() / RAND_MAX);
257
258         random_wave_number_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
259
260         random_frequency_vec[i] = ((double)rand() / RAND_MAX);
261
262         random_direction_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
263
264         random_phase_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
265     }
266
267     // 2. generate noise vec
268     double amp = 0;
269     double wave_no = 0;
270     double freq = 0;
271     double dir = 0;
272     double phase = 0;
273
274     double x = 0;
275     double y = 0;
276     double t = time(NULL);
277
278     double max_noise = -1 * std::numeric_limits<double>::infinity();
279     double min_noise = std::numeric_limits<double>::infinity();
280
281     double noise = 0;
282     std::vector<double> noise_vec(n_elements, 0);
283
284     for (int i = 0; i < n_elements; i++) {
285         x = this->tile_position_x_vec[i] - this->position_x;
286         y = this->tile_position_y_vec[i] - this->position_y;
287
288         for (int j = 0; j < n_components; j++) {
289             amp = random_amplitude_vec[j];
290             wave_no = random_wave_number_vec[j];
291             freq = random_frequency_vec[j];
292             dir = random_direction_vec[j];
293             phase = random_phase_vec[j];
294
295             noise += (amp / (j + 1)) * cos(
296                 wave_no * (j + 1) * (x * sin(dir) + y * cos(dir)) +
297                 2 * M_PI * (j + 1) * freq * t +
298                 phase
299             );
300         }
301
302         noise_vec[i] = noise;
303
304         if (noise > max_noise) {
```

```

305         max_noise = noise;
306     }
307
308     else if (noise < min_noise) {
309         min_noise = noise;
310     }
311
312     noise = 0;
313 }
314
315 // 3. normalize noise vec
316 for (int i = 0; i < n_elements; i++) {
317     noise_vec[i] = (noise_vec[i] - min_noise) / (max_noise - min_noise);
318
319     if (noise_vec[i] < 0) {
320         noise_vec[i] = 0;
321     }
322     else if (noise_vec[i] > 1) {
323         noise_vec[i] = 1;
324     }
325 }
326
327 return noise_vec;
328 } /* __getNoise() */

```

3.3.3.6 __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).

```

793 {
794     HexTile* selected_tile_ptr = NULL;
795
796     bool break_flag = false;
797     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
798     std::map<double, HexTile*>::iterator hex_map_iter_y;
799
800     for (
801         hex_map_iter_x = this->hex_map.begin();
802         hex_map_iter_x != this->hex_map.end();
803         hex_map_iter_x++
804     ) {
805         for (
806             hex_map_iter_y = hex_map_iter_x->second.begin();
807             hex_map_iter_y != hex_map_iter_x->second.end();
808             hex_map_iter_y++
809         ) {
810             if (hex_map_iter_y->second->is_selected) {
811                 selected_tile_ptr = hex_map_iter_y->second;
812                 break_flag = true;
813             }
814
815             if (break_flag) {
816                 break;
817             }
818         }
819
820         if (break_flag) {
821             break;
822         }
823     }
824
825     return selected_tile_ptr;
826 } /* __getSelectedTile() */

```


3.3.3.7 __getValidMapIndexPositions()

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

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

Parameters

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

Returns

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

```
413 {
414     std::vector<double> map_index_positions = {-1, -1};
415
416     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
417     std::map<double, HexTile*>::iterator hex_map_iter_y;
418     HexTile* hex_ptr;
419
420     double distance = 0;
421
422     for (
423         hex_map_iter_x = this->hex_map.begin();
424         hex_map_iter_x != this->hex_map.end();
425         hex_map_iter_x++
426     ) {
427         for (
428             hex_map_iter_y = hex_map_iter_x->second.begin();
429             hex_map_iter_y != hex_map_iter_x->second.end();
430             hex_map_iter_y++
431         ) {
432             hex_ptr = hex_map_iter_y->second;
433
434             distance = sqrt(
435                 pow(hex_ptr->position_x - potential_x, 2) +
436                 pow(hex_ptr->position_y - potential_y, 2)
437             );
438
439             if (distance <= hex_ptr->minor_radius / 4) {
440                 map_index_positions = {hex_ptr->position_x, hex_ptr->position_y};
441                 return map_index_positions;
442             }
443         }
444     }
445
446     return map_index_positions;
447 } /* __isInHexMap() */
```

3.3.3.8 __isLakeTouchingOcean()

```
bool HexMap::__isLakeTouchingOcean (
    HexTile * hex_ptr ) [private]
636 {
637     // 1. if not lake tile, return
638     if (not (hex_ptr->tile_type == TileType :: LAKE)) {
639         return false;
640     }
641
642     // 2. scan neighbours for ocean tiles
643     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
```

```

644
645     for (size_t i = 0; i < neighbours_vec.size(); i++) {
646         if (neighbours_vec[i]->tile_type == TileType::OCEAN) {
647             return true;
648         }
649     }
650
651     return false;
652 } /* __isLakeTouchingOcean() */

```

3.3.3.9 __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->assets_manager_ptr,
62         this->inputs_handler_ptr,
63         this->messages_handler_ptr,
64         this->render_window_ptr
65     );
66
67     this->hex_map[this->position_x][this->position_y] = hex_ptr;
68     this->tile_position_x_vec.push_back(hex_ptr->position_x);
69     this->tile_position_y_vec.push_back(hex_ptr->position_y);
70     this->n_tiles++;
71
72
73     // 2. fill out first row (reflect across origin tile)
74     for (int i = 0; i < this->n_layers; i++) {
75         hex_ptr = new HexTile(
76             this->position_x + 2 * (i + 1) * hex_ptr->minor_radius,
77             this->position_y,
78             this->assets_manager_ptr,
79             this->inputs_handler_ptr,
80             this->messages_handler_ptr,
81             this->render_window_ptr
82         );
83
84         this->hex_map[this->position_x][this->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->assets_manager_ptr,
97             this->inputs_handler_ptr,
98             this->messages_handler_ptr,
99             this->render_window_ptr
100        );
101
102        this->hex_map[this->position_x][this->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->assets_manager_ptr,
139             this->inputs_handler_ptr,
140             this->messages_handler_ptr,
141             this->render_window_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->assets_manager_ptr,
158                 this->inputs_handler_ptr,
159                 this->messages_handler_ptr,
160                 this->render_window_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->assets_manager_ptr,
186             this->inputs_handler_ptr,
187             this->messages_handler_ptr,
188             this->render_window_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->assets_manager_ptr,
205         this->inputs_handler_ptr,
206         this->messages_handler_ptr,
207         this->render_window_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() */

```

3.3.3.10 __procedurallyGenerateTileResources()

```

void HexMap::__procedurallyGenerateTileResources (
    void ) [private]

```

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

```

718 {
719     // 1. get random cosine series noise vec
720     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
721
722     // 2. set tile resources based on random cosine series noise
723     int noise_idx = 0;
724
725     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
726     std::map<double, HexTile*>::iterator hex_map_iter_y;
727     for (
728         hex_map_iter_x = this->hex_map.begin();
729         hex_map_iter_x != this->hex_map.end();
730         hex_map_iter_x++
731     ) {
732         for (
733             hex_map_iter_y = hex_map_iter_x->second.begin();
734             hex_map_iter_y != hex_map_iter_x->second.end();
735             hex_map_iter_y++
736         ) {
737             hex_map_iter_y->second->setTileResource(noise_vec[noise_idx]);
738             noise_idx++;
739         }
740     }
741
742     return;
743 } /* __procedurallyGenerateTileResources() */

```

3.3.3.11 __procedurallyGenerateTileTypes()

```

void HexMap::__procedurallyGenerateTileTypes (
    void ) [private]

```

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

```

343 {
344     // 1. get random cosine series noise vec
345     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
346
347     // 2. set initial tile types based on either random cosine series noise or white
348     //     noise (decided by coin toss)
349     int noise_idx = 0;

```

```

350
351     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
352     std::map<double, HexTile*>::iterator hex_map_iter_y;
353     for (
354         hex_map_iter_x = this->hex_map.begin();
355         hex_map_iter_x != this->hex_map.end();
356         hex_map_iter_x++
357     ) {
358         for (
359             hex_map_iter_y = hex_map_iter_x->second.begin();
360             hex_map_iter_y != hex_map_iter_x->second.end();
361             hex_map_iter_y++
362         ) {
363             if ((double)rand() / RAND_MAX > 0.5) {
364                 hex_map_iter_y->second->setTileType(noise_vec[noise_idx]);
365             }
366             else {
367                 hex_map_iter_y->second->setTileType((double)rand() / RAND_MAX);
368             }
369             noise_idx++;
370         }
371     }
372
373     // 3. smooth tile types (majority rules)
374     this->__smoothTileTypes();
375
376     // 4. set border tile type to ocean
377     for (size_t i = 0; i < this->border_tiles_vec.size(); i++) {
378         this->border_tiles_vec[i]->setTileType(TileType :: OCEAN);
379     }
380
381     // 5. enforce ocean continuity (i.e. all lake tiles touching ocean become ocean)
382     this->__enforceOceanContinuity();
383
384     return;
385 } /* __procedurallyGenerateTileTypes() */

```

3.3.3.12 __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(40, 40, 40, 40));
37
38     return;
39 } /* __setUpGlassScreen() */

```

3.3.3.13 __smoothTileTypes()

```

void HexMap::__smoothTileTypes (
    void ) [private]

```

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

```

589 {
590     std::cout << "smoothing ..." << std::endl;
591
592     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
593     std::map<double, HexTile*>::iterator hex_map_iter_y;
594     HexTile* hex_ptr;
595     TileType majority_tile_type;
596
597     for (
598         hex_map_iter_x = this->hex_map.begin();
599         hex_map_iter_x != this->hex_map.end();
600         hex_map_iter_x++
601     ) {

```

```

602         for (
603             hex_map_iter_y = hex_map_iter_x->second.begin();
604             hex_map_iter_y != hex_map_iter_x->second.end();
605             hex_map_iter_y++
606         ) {
607             hex_ptr = hex_map_iter_y->second;
608             majority_tile_type = this->__getMajorityTileType(hex_ptr);
609
610             if (majority_tile_type != hex_ptr->tile_type) {
611                 hex_ptr->setTileType(majority_tile_type);
612             }
613         }
614     }
615
616     return;
617 } /* __smoothTileTypes() */

```

3.3.3.14 assess()

```

void HexMap::assess (
    void )

```

Method to assess the resource of the selected tile.

```

908 {
909     HexTile* selected_tile_ptr = this->__getSelectedTile();
910     if (selected_tile_ptr != NULL) {
911         selected_tile_ptr->assess();
912     }
913
914     return;
915 } /* assess() */

```

3.3.3.15 clear()

```

void HexMap::clear (
    void )

```

Method to clear the hex map.

```

1064 {
1065     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1066     std::map<double, HexTile*>::iterator hex_map_iter_y;
1067     for (
1068         hex_map_iter_x = this->hex_map.begin();
1069         hex_map_iter_x != this->hex_map.end();
1070         hex_map_iter_x++
1071     ) {
1072         for (
1073             hex_map_iter_y = hex_map_iter_x->second.begin();
1074             hex_map_iter_y != hex_map_iter_x->second.end();
1075             hex_map_iter_y++
1076         ) {
1077             delete hex_map_iter_y->second;
1078         }
1079     }
1080     this->hex_map.clear();
1081
1082     this->tile_position_x_vec.clear();
1083     this->tile_position_y_vec.clear();
1084     this->border_tiles_vec.clear();
1085
1086     return;
1087 } /* clear() */

```

3.3.3.16 draw()

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

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

```
1020 {
1021     // 1. draw all tiles in order
1022     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
1023     std::map<double, HexTile*>::iterator hex_map_iter_y;
1024     for (
1025         hex_map_iter_x = this->hex_map.begin();
1026         hex_map_iter_x != this->hex_map.end();
1027         hex_map_iter_x++
1028     ) {
1029         for (
1030             hex_map_iter_y = hex_map_iter_x->second.begin();
1031             hex_map_iter_y != hex_map_iter_x->second.end();
1032             hex_map_iter_y++
1033         ) {
1034             hex_map_iter_y->second->draw();
1035         }
1036     }
1037
1038     // 2. redraw selected tile
1039     HexTile* selected_tile_ptr = this->__getSelectedTile();
1040     if (selected_tile_ptr != NULL) {
1041         selected_tile_ptr->draw();
1042     }
1043
1044     // 3. draw glass screen
1045     this->render_window_ptr->draw(this->glass_screen);
1046
1047     this->frame++;
1048     return;
1049 } /* draw() */
```

3.3.3.17 process()

```
void HexMap::process (
    void )
```

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

```
930 {
931     // 1. handle inputs
932     //...
933
934     // 2. process tiles
935     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
936     std::map<double, HexTile*>::iterator hex_map_iter_y;
937     for (
938         hex_map_iter_x = this->hex_map.begin();
939         hex_map_iter_x != this->hex_map.end();
940         hex_map_iter_x++
941     ) {
942         for (
943             hex_map_iter_y = hex_map_iter_x->second.begin();
944             hex_map_iter_y != hex_map_iter_x->second.end();
945             hex_map_iter_y++
946         ) {
947             hex_map_iter_y->second->process();
948         }
949     }
950
951     return;
952 } /* process() */
```

3.3.3.18 reroll()

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

Method to re-roll the hex map.

```
967 {
968     this->clear();
969     this->__assembleHexMap();
970
971     return;
972 } /* reroll() */
```

3.3.3.19 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
987 {
988     std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
989     std::map<double, HexTile*>::iterator hex_map_iter_y;
990     for (
991         hex_map_iter_x = this->hex_map.begin();
992         hex_map_iter_x != this->hex_map.end();
993         hex_map_iter_x++
994     ) {
995         for (
996             hex_map_iter_y = hex_map_iter_x->second.begin();
997             hex_map_iter_y != hex_map_iter_x->second.end();
998             hex_map_iter_y++
999         ) {
1000             hex_map_iter_y->second->toggleResourceOverlay();
1001         }
1002     }
1003
1004     return;
1005 } /* toggleResourceOverlay() */
```

3.3.4 Member Data Documentation

3.3.4.1 assets_manager_ptr

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

A pointer to the assets manager.

3.3.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

3.3.4.3 frame

```
int HexMap::frame
```

The current frame of this object.

3.3.4.4 glass_screen

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

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

3.3.4.5 hex_map

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

A position-indexed, nested map of hex tiles.

3.3.4.6 inputs_handler_ptr

```
InputsHandler* HexMap::inputs_handler_ptr [private]
```

A pointer to the inputs handler.

3.3.4.7 messages_handler_ptr

```
MessagesHandler* HexMap::messages_handler_ptr [private]
```

A pointer to the messages handler.

3.3.4.8 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

3.3.4.9 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

3.3.4.10 position_x

```
double HexMap::position_x
```

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

3.3.4.11 position_y

```
double HexMap::position_y
```

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

3.3.4.12 render_window_ptr

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

A pointer to the render window.

3.3.4.13 tile_position_x_vec

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

A vector of tile x positions.

3.3.4.14 tile_position_y_vec

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

A vector of tile y position.

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

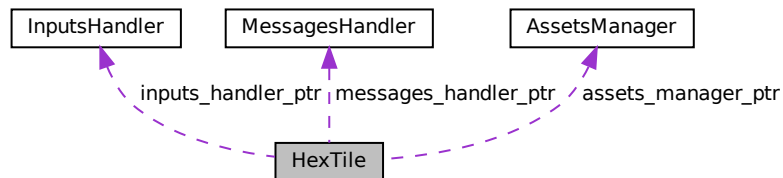
- header/HexMap/[HexMap.h](#)
- source/HexMap/[HexMap.cpp](#)

3.4 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, [AssetsManager](#) *, [InputsHandler](#) *, [MessagesHandler](#) *, sf::RenderWindow *)
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 [toggleResourceOverlay](#) (void)
Method to toggle the tile resource overlay.
- void [assess](#) (void)
Method to assess the tile's resource.
- void [process](#) (void)
Method to process [HexTile](#). To be called once per frame.
- 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.
- int [frame](#)
The current frame of this object.
- 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.

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.

Private Attributes

- [AssetsManager](#) * [assets_manager_ptr](#)
A pointer to the assets manager.
- [InputsHandler](#) * [inputs_handler_ptr](#)
A pointer to the inputs handler.
- [MessagesHandler](#) * [messages_handler_ptr](#)
A pointer to the messages handler.
- sf::RenderWindow * [render_window_ptr](#)
A pointer to the render window.

3.4.1 Detailed Description

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

3.4.2 Constructor & Destructor Documentation

3.4.2.1 HexTile()

```
HexTile::HexTile (
    double position_x,
    double position_y,
    AssetsManager * assets_manager_ptr,
    InputsHandler * inputs_handler_ptr,
    MessagesHandler * messages_handler_ptr,
    sf::RenderWindow * render_window_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>assets_manager_ptr</i>	Pointer to the assets manager.
<i>inputs_handler_ptr</i>	Pointer to the inputs handler.
<i>messages_handler_ptr</i>	Pointer to the messages handler.
<i>render_window_ptr</i>	Pointer to the render window.

```
300 {
301     // 1. set attributes
302     this->assets_manager_ptr = assets_manager_ptr;
303     this->inputs_handler_ptr = inputs_handler_ptr;
304     this->messages_handler_ptr = messages_handler_ptr;
305     this->render_window_ptr = render_window_ptr;
306
307     this->show_node = false;
308     this->show_resource = false;
309     this->resource_assessed = false;
310     this->is_selected = false;
311
312     this->frame = 0;
313
314     this->position_x = position_x;
315     this->position_y = position_y;
316
317     this->major_radius = 32;
318     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
319
320     // 2. set up and position drawable attributes
321     this->__setUpNodeSprite();
322     this->__setUpTileSprite();
323     this->__setUpSelectOutlineSprite();
324     this->__setUpResourceChipSprite();
325     this->__setUpResourceText();
326
327     // 3. set tile type and resource (default to forest and average)
328     this->setTileType(TileType :: FOREST);
329     this->setTileResource(TileResource :: AVERAGE);
330 }
```

```

331     std::cout << "HexTile constructed at " << this << std::endl;
332
333     return;
334 } /* HexTile() */

```

3.4.2.2 ~HexTile()

```

HexTile::~HexTile (
    void )

```

Destructor for the [HexTile](#) class.

```

655 {
656     std::cout << "HexTile at " << this << " destroyed" << std::endl;
657
658     return;
659 } /* ~HexTile() */

```

3.4.3 Member Function Documentation

3.4.3.1 __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.

```

236 {
237     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
238
239     double mouse_x = mouse_position.x;
240     double mouse_y = mouse_position.y;
241
242     double distance = sqrt(
243         pow(this->position_x - mouse_x, 2) +
244         pow(this->position_y - mouse_y, 2)
245     );
246
247     if (distance < this->minor_radius) {
248         return true;
249     }
250     else {
251         return false;
252     }
253 } /* __isClicked() */

```

3.4.3.2 __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     switch (this->tile_resource) {
163         case (TileResource :: POOR): {
164             this->resource_text.setString("-2");
165
166             break;
167         }
168
169         case (TileResource :: BELOW_AVERAGE): {
170             this->resource_text.setString("-1");
171
172             break;
173         }
174
175         case (TileResource :: AVERAGE): {
176             this->resource_text.setString("0");
177
178             break;
179         }
180
181         case (TileResource :: ABOVE_AVERAGE): {
182             this->resource_text.setString("+1");
183
184             break;
185         }
186
187         case (TileResource :: GOOD): {
188             this->resource_text.setString("+2");
189
190             break;
191         }
192
193         default: {
194             this->resource_text.setString("?");
195
196             break;
197         }
198     }
199
200     if (not this->resource_assessed) {
201         this->resource_text.setString("?");
202     }
203
204     this->resource_text.setCharacterSize(16);
205
206     this->resource_text.setOrigin(
207         this->resource_text.getLocalBounds().width / 2,
208         this->resource_text.getLocalBounds().height / 2
209     );
210
211     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
212
213     this->resource_text.setPosition(
214         this->position_x,
215         this->position_y - 4
216     );
217
218     return;
219 } /* __setResourceText() */
```

3.4.3.3 __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() */

```

3.4.3.4 __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(sf::Color(175, 175, 175, 175));
143
144     return;
145 } /* __setUpResourceChip() */

```

3.4.3.5 __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(MONOCROME_TEXT_RED);
113
114     this->select_outline_sprite.setFillColor(sf::Color(0, 0, 0, 0));
115
116     return;
117 } /* __setUpSelectOutline() */

```


3.4.3.6 __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() */
```

3.4.3.7 assess()

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

Method to assess the tile's resource.

```
555 {
556     this->resource_assessed = true;
557     this->__setResourceText();
558
559     return;
560 } /* assess() */
```

3.4.3.8 draw()

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

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

```
611 {
612     // 1. draw hex
613     this->render_window_ptr->draw(this->tile_sprite);
614
615     // 2. draw node
616     if (this->show_node) {
617         this->render_window_ptr->draw(this->node_sprite);
618     }
619
620     // 3. draw resource
621     if (this->show_resource) {
622         this->render_window_ptr->draw(this->resource_chip_sprite);
623         this->render_window_ptr->draw(this->resource_text);
624     }
625
626     // 4. draw selection outline
627     if (this->is_selected) {
628         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
629
630         outline_colour.a =
```

```

631         255 * pow(cos((M_PI * this->frame) / (1.5 * FRAMES_PER_SECOND)), 2);
632
633         this->select_outline_sprite.setOutlineColor(outline_colour);
634
635         this->render_window_ptr->draw(this->select_outline_sprite);
636     }
637
638     this->frame++;
639     return;
640 } /* draw() */

```

3.4.3.9 process()

```

void HexTile::process (
    void )

```

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

```

575 {
576     // 1. handle inputs
577     if (inputs_handler_ptr->mouse_left_click) {
578         if (this->__isClicked()) {
579             std::cout << "Tile (" << this->position_x << ", " << this->position_y <<
580                 ") was selected" << std::endl;
581
582             this->is_selected = true;
583         }
584
585         else {
586             this->is_selected = false;
587         }
588     }
589
590     if (inputs_handler_ptr->mouse_right_click) {
591         this->is_selected = false;
592     }
593
594     return;
595 } /* process() */

```

3.4.3.10 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].
--------------------	--

```

480 {
481     // 1. check input
482     if (input_value < 0 or input_value > 1) {
483         std::string error_str = "ERROR HexTile::setTileResource() given input value is ";
484         error_str += "not in the closed interval [0, 1]";
485
486         #ifdef _WIN32
487             std::cout << error_str << std::endl;
488         #endif /* _WIN32 */
489
490         throw std::runtime_error(error_str);
491     }
492
493     // 2. convert input value to tile resource
494     TileResource tile_resource;

```

```

495
496     if (input_value <= tile_resource_cumulative_probabilities[0]) {
497         tile_resource = TileResource :: POOR;
498     }
499     else if (input_value <= tile_resource_cumulative_probabilities[1]) {
500         tile_resource = TileResource :: BELOW_AVERAGE;
501     }
502     else if (input_value <= tile_resource_cumulative_probabilities[2]) {
503         tile_resource = TileResource :: AVERAGE;
504     }
505     else if (input_value <= tile_resource_cumulative_probabilities[3]) {
506         tile_resource = TileResource :: ABOVE_AVERAGE;
507     }
508     else {
509         tile_resource = TileResource :: GOOD;
510     }
511
512     // 3. call alternate method
513     this->setTileResource(tile_resource);
514
515     return;
516 } /* setTileResource(double) */

```

3.4.3.11 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.
----------------------	---

```

458 {
459     this->tile_resource = tile_resource;
460     this->__setResourceText();
461
462     return;
463 } /* setTileResource(TileResource) */

```

3.4.3.12 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].
--------------------	--

```

408 {
409     // 1. check input
410     if (input_value < 0 or input_value > 1) {
411         std::string error_str = "ERROR HexTile::setTileType() given input value is ";
412         error_str += "not in the closed interval [0, 1]";
413
414         #ifdef _WIN32
415             std::cout << error_str << std::endl;
416         #endif /* _WIN32 */
417

```

```

418         throw std::runtime_error(error_str);
419     }
420
421     // 2. convert input value to tile type
422     TileType tile_type;
423
424     if (input_value <= tile_type_cumulative_probabilities[0]) {
425         tile_type = TileType :: LAKE;
426     }
427     else if (input_value <= tile_type_cumulative_probabilities[1]) {
428         tile_type = TileType :: PLAINS;
429     }
430     else if (input_value <= tile_type_cumulative_probabilities[2]) {
431         tile_type = TileType :: FOREST;
432     }
433     else {
434         tile_type = TileType :: MOUNTAINS;
435     }
436
437     // 3. call alternate method
438     this->setTileType(tile_type);
439
440     return;
441 } /* setTileType(double) */

```

3.4.3.13 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.
------------------	---

```

349 {
350     this->tile_type = tile_type;
351
352     switch (this->tile_type) {
353         case (TileType :: FOREST): {
354             this->tile_sprite.setFillColor(FOREST_GREEN);
355
356             break;
357         }
358
359         case (TileType :: LAKE): {
360             this->tile_sprite.setFillColor(LAKE_BLUE);
361
362             break;
363         }
364
365         case (TileType :: MOUNTAINS): {
366             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
367
368             break;
369         }
370
371         case (TileType :: OCEAN): {
372             this->tile_sprite.setFillColor(OCEAN_BLUE);
373
374             break;
375         }
376
377         case (TileType :: PLAINS): {
378             this->tile_sprite.setFillColor(PLAINS_YELLOW);
379
380             break;
381         }
382
383         default: {
384             // do nothing!
385
386             break;
387         }

```

```
388     }
389
390     return;
391 } /* setTileType(TileType) */
```

3.4.3.14 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
531 {
532     if (this->show_resource) {
533         this->show_resource = false;
534     }
535     else {
536         this->show_resource = true;
537     }
538
539     return;
540 } /* toggleResourceOverlay() */
```

3.4.4 Member Data Documentation

3.4.4.1 assets_manager_ptr

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

A pointer to the assets manager.

3.4.4.2 frame

```
int HexTile::frame
```

The current frame of this object.

3.4.4.3 inputs_handler_ptr

```
InputsHandler* HexTile::inputs_handler_ptr [private]
```

A pointer to the inputs handler.

3.4.4.4 is_selected

```
bool HexTile::is_selected
```

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

3.4.4.5 major_radius

```
double HexTile::major_radius
```

The radius of the smallest bounding circle.

3.4.4.6 messages_handler_ptr

```
MessagesHandler* HexTile::messages_handler_ptr [private]
```

A pointer to the messages handler.

3.4.4.7 minor_radius

```
double HexTile::minor_radius
```

The radius of the largest inscribed circle.

3.4.4.8 node_sprite

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

A circle shape to mark the tile node.

3.4.4.9 position_x

```
double HexTile::position_x
```

The x position of the tile.

3.4.4.10 position_y

```
double HexTile::position_y
```

The y position of the tile.

3.4.4.11 render_window_ptr

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

A pointer to the render window.

3.4.4.12 resource_assessed

```
bool HexTile::resource_assessed
```

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

3.4.4.13 resource_chip_sprite

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

A circle shape which represents a resource chip.

3.4.4.14 resource_text

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

A text representation of the resource.

3.4.4.15 select_outline_sprite

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

A convex shape which outlines the tile when selected.

3.4.4.16 show_node

```
bool HexTile::show_node
```

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

3.4.4.17 show_resource

```
bool HexTile::show_resource
```

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

3.4.4.18 tile_resource

```
TileResource HexTile::tile_resource
```

3.4.4.19 tile_sprite

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

A convex shape which represents the tile.

3.4.4.20 tile_type

```
TileType HexTile::tile_type
```

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

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

3.5 InputsHandler Class Reference

A class which handles inputs from peripherals (i.e., keyboard and mouse).

```
#include <InputsHandler.h>
```


Public Member Functions

- [InputHandler](#) (void)
Constructor for the [InputHandler](#) class.
- void [process](#) (sf::Event *)
- void [printKeysPressed](#) (void)
Method to print out which keys are currently pressed.
- void [reset](#) (void)
Method to reset [InputHandler](#). To be called once per frame (at end of frame!).
- [~InputHandler](#) (void)
Destructor for the [InputHandler](#) class.

Public Attributes

- bool [mouse_left_click](#)
A boolean which indicates if the mouse left button has been clicked.
- bool [mouse_right_click](#)
A boolean which indicates if the mouse right button has been clicked.
- std::vector< bool > [key_pressed_once_vec](#)
A vector (bool) which indicates which keys have been pressed once. Useful for discrete inputs.
- std::vector< bool > [key_press_vec](#)
A vector <bool> which indicates which keys are currently pressed. Useful for smooth movement.
- std::map< sf::Keyboard::Key, std::string > [key_code_map](#)
A map from key codes to corresponding string representations.

Private Member Functions

- void [__constructKeyCodeMap](#) (void)
Helper method to construct a map from sf::Keyboard::Key to a string representation of the corresponding key.

3.5.1 Detailed Description

A class which handles inputs from peripherals (i.e., keyboard and mouse).

3.5.2 Constructor & Destructor Documentation

3.5.2.1 InputHandler()

```
InputHandler::InputHandler (
    void )
```

Constructor for the [InputHandler](#) class.

```
379 {
380     this->key_pressed_once_vec.resize(sf::Keyboard::KeyCount, false);
381     this->key_press_vec.resize(sf::Keyboard::KeyCount, false);
382
383     this->__constructKeyCodeMap();
384
385     std::cout << "InputHandler constructed at " << this << std::endl;
386
387     return;
388 } /* InputHandler() */
```

3.5.2.2 ~InputsHandler()

```
InputsHandler::~InputsHandler (
    void )
```

Destructor for the [InputsHandler](#) class.

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

3.5.3 Member Function Documentation

3.5.3.1 __constructKeyCodeMap()

```
void InputsHandler::__constructKeyCodeMap (
    void ) [private]
```

Helper method to construct a map from sf::Keyboard::Key to a string representation of the corresponding key.

```
35 {
36     // 1. unknown keys
37     this->key_code_map.insert(
38         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Unknown, "Unknown")
39     );
40
41
42     // 2. alpha keys
43     this->key_code_map.insert(
44         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::A, "A")
45     );
46     this->key_code_map.insert(
47         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::B, "B")
48     );
49     this->key_code_map.insert(
50         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::C, "C")
51     );
52     this->key_code_map.insert(
53         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::D, "D")
54     );
55     this->key_code_map.insert(
56         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::E, "E")
57     );
58     this->key_code_map.insert(
59         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F, "F")
60     );
61     this->key_code_map.insert(
62         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::G, "G")
63     );
64     this->key_code_map.insert(
65         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::H, "H")
66     );
67     this->key_code_map.insert(
68         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::I, "I")
69     );
70     this->key_code_map.insert(
71         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::J, "J")
72     );
73     this->key_code_map.insert(
74         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::K, "K")
75     );
76     this->key_code_map.insert(
77         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::L, "L")
78     );
79     this->key_code_map.insert(
80         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::M, "M")
81     );
82     this->key_code_map.insert(
83         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::N, "N")
84     );
85     this->key_code_map.insert(
```

```
86         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::O, "O")
87     );
88     this->key_code_map.insert(
89         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::P, "P")
90     );
91     this->key_code_map.insert(
92         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Q, "Q")
93     );
94     this->key_code_map.insert(
95         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::R, "R")
96     );
97     this->key_code_map.insert(
98         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::S, "S")
99     );
100    this->key_code_map.insert(
101        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::T, "T")
102    );
103    this->key_code_map.insert(
104        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::U, "U")
105    );
106    this->key_code_map.insert(
107        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::V, "V")
108    );
109    this->key_code_map.insert(
110        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::W, "W")
111    );
112    this->key_code_map.insert(
113        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::X, "X")
114    );
115    this->key_code_map.insert(
116        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Y, "Y")
117    );
118    this->key_code_map.insert(
119        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Z, "Z")
120    );
121
122
123    // 3. numeric keys
124    this->key_code_map.insert(
125        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num0, "0")
126    );
127    this->key_code_map.insert(
128        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num1, "1")
129    );
130    this->key_code_map.insert(
131        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num2, "2")
132    );
133    this->key_code_map.insert(
134        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num3, "3")
135    );
136    this->key_code_map.insert(
137        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num4, "4")
138    );
139    this->key_code_map.insert(
140        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num5, "5")
141    );
142    this->key_code_map.insert(
143        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num6, "6")
144    );
145    this->key_code_map.insert(
146        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num7, "7")
147    );
148    this->key_code_map.insert(
149        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num8, "8")
150    );
151    this->key_code_map.insert(
152        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num9, "9")
153    );
154    this->key_code_map.insert(
155        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad0, "0")
156    );
157    this->key_code_map.insert(
158        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad1, "1")
159    );
160    this->key_code_map.insert(
161        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad2, "2")
162    );
163    this->key_code_map.insert(
164        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad3, "3")
165    );
166    this->key_code_map.insert(
167        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad4, "4")
168    );
169    this->key_code_map.insert(
170        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad5, "5")
171    );
172    this->key_code_map.insert(
```

```

173         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad6, "6")
174     );
175     this->key_code_map.insert (
176         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad7, "7")
177     );
178     this->key_code_map.insert (
179         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad8, "8")
180     );
181     this->key_code_map.insert (
182         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad9, "9")
183     );
184
185
186     // 4. direction keys
187     this->key_code_map.insert (
188         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Left, "Left")
189     );
190     this->key_code_map.insert (
191         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Right, "Right")
192     );
193     this->key_code_map.insert (
194         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Up, "Up")
195     );
196     this->key_code_map.insert (
197         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Down, "Down")
198     );
199
200
201     // 5. function keys
202     this->key_code_map.insert (
203         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F1, "F1")
204     );
205     this->key_code_map.insert (
206         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F2, "F2")
207     );
208     this->key_code_map.insert (
209         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F3, "F3")
210     );
211     this->key_code_map.insert (
212         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F4, "F4")
213     );
214     this->key_code_map.insert (
215         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F5, "F5")
216     );
217     this->key_code_map.insert (
218         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F6, "F6")
219     );
220     this->key_code_map.insert (
221         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F7, "F7")
222     );
223     this->key_code_map.insert (
224         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F8, "F8")
225     );
226     this->key_code_map.insert (
227         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F9, "F9")
228     );
229     this->key_code_map.insert (
230         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F10, "F10")
231     );
232     this->key_code_map.insert (
233         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F11, "F11")
234     );
235     this->key_code_map.insert (
236         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F12, "F12")
237     );
238     this->key_code_map.insert (
239         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F13, "F13")
240     );
241     this->key_code_map.insert (
242         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F14, "F14")
243     );
244     this->key_code_map.insert (
245         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F15, "F15")
246     );
247
248
249     // 6. other keys
250     this->key_code_map.insert (
251         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Escape, "Escape")
252     );
253     this->key_code_map.insert (
254         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LControl, "LCtrl")
255     );
256     this->key_code_map.insert (
257         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LShift, "LShift")
258     );
259     this->key_code_map.insert (

```

```

260         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LAlt, "LAlt")
261     );
262     this->key_code_map.insert (
263         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LSystem, "LSystem")
264     );
265     this->key_code_map.insert (
266         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RControl, "RCtrl")
267     );
268     this->key_code_map.insert (
269         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RShift, "RShift")
270     );
271     this->key_code_map.insert (
272         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RAlt, "RAlt")
273     );
274     this->key_code_map.insert (
275         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RSystem, "RSystem")
276     );
277     this->key_code_map.insert (
278         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Menu, "Menu")
279     );
280     this->key_code_map.insert (
281         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LBracket, "LBracket")
282     );
283     this->key_code_map.insert (
284         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RBracket, "RBracket")
285     );
286     this->key_code_map.insert (
287         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Semicolon, "Semicolon")
288     );
289     this->key_code_map.insert (
290         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Comma, "Comma")
291     );
292     this->key_code_map.insert (
293         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Period, "Period")
294     );
295     this->key_code_map.insert (
296         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Quote, "Quote")
297     );
298     this->key_code_map.insert (
299         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Slash, "Slash")
300     );
301     this->key_code_map.insert (
302         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Backslash, "Backslash")
303     );
304     this->key_code_map.insert (
305         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Tilde, "Tilde")
306     );
307     this->key_code_map.insert (
308         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Equal, "Equal")
309     );
310     this->key_code_map.insert (
311         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Hyphen, "Hyphen")
312     );
313     this->key_code_map.insert (
314         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Space, "Space")
315     );
316     this->key_code_map.insert (
317         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Enter, "Enter")
318     );
319     this->key_code_map.insert (
320         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Backspace, "Backspace")
321     );
322     this->key_code_map.insert (
323         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Tab, "Tab")
324     );
325     this->key_code_map.insert (
326         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::PageUp, "PageUp")
327     );
328     this->key_code_map.insert (
329         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::PageDown, "PageDown")
330     );
331     this->key_code_map.insert (
332         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::End, "End")
333     );
334     this->key_code_map.insert (
335         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Home, "Home")
336     );
337     this->key_code_map.insert (
338         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Insert, "Insert")
339     );
340     this->key_code_map.insert (
341         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Delete, "Delete")
342     );
343     this->key_code_map.insert (
344         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Add, "Add")
345     );
346     this->key_code_map.insert (

```

```

347         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Subtract, "Subtract")
348     );
349     this->key_code_map.insert (
350         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Multiply, "Multiply")
351     );
352     this->key_code_map.insert (
353         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Divide, "Divide")
354     );
355     this->key_code_map.insert (
356         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Pause, "Pause")
357     );
358
359     return;
360 } /* __constructKeyCodeMap() */

```

3.5.3.2 printKeysPressed()

```

void InputsHandler::printKeysPressed (
    void )

```

Method to print out which keys are currently pressed.

```

473 {
474     std::string print_str = "";
475
476     for (size_t i = 0; i < this->key_press_vec.size(); i++) {
477         if (this->key_press_vec[i]) {
478             print_str += this->key_code_map[sf::Keyboard::Key(i)];
479             print_str += ", ";
480         }
481     }
482
483     if (not print_str.empty()) {
484         std::cout << "Keys pressed: " << print_str << std::endl;
485     }
486
487     return;
488 } /* printKeysPressed() */

```

3.5.3.3 process()

```

void InputsHandler::process (
    sf::Event * event_ptr )
405 {
406     // 1. update state of key press vectors
407     switch (event_ptr->type) {
408         case (sf::Event::KeyPressed): {
409             if (not this->key_press_vec[event_ptr->key.code]) {
410                 this->key_pressed_once_vec[event_ptr->key.code] = true;
411             }
412
413             this->key_press_vec[event_ptr->key.code] = true;
414
415             break;
416         }
417
418         case (sf::Event::KeyReleased): {
419             this->key_pressed_once_vec[event_ptr->key.code] = false;
420             this->key_press_vec[event_ptr->key.code] = false;
421
422             break;
423         }
424
425         case (sf::Event::MouseButtonPressed): {
426             if (sf::Mouse::isButtonPressed(sf::Mouse::Left))
427             {
428                 this->mouse_left_click = true;
429
430                 std::cout << "left click" << std::endl;
431             }
432

```

```

433         if (sf::Mouse::isButtonPressed(sf::Mouse::Right))
434         {
435             this->mouse_right_click = true;
436
437             std::cout << "right click" << std::endl;
438         }
439
440         break;
441     }
442
443     case (sf::Event::MouseButtonReleased): {
444         this->mouse_left_click = false;
445         this->mouse_right_click = false;
446
447         break;
448     }
449
450     default: {
451         // do nothing!
452
453         break;
454     }
455 }
456
457 return;
458 } /* process() */

```

3.5.3.4 reset()

```

void InputsHandler::reset (
    void )

```

Method to reset [InputsHandler](#). To be called once per frame (at end of frame!).

```

503 {
504     this->mouse_left_click = false;
505     this->mouse_right_click = false;
506
507     for (size_t i = 0; i < this->key_press_vec.size(); i++) {
508         this->key_pressed_once_vec[i] = false;
509     }
510
511     return;
512 } /* reset() */

```

3.5.4 Member Data Documentation

3.5.4.1 key_code_map

```
std::map<sf::Keyboard::Key, std::string> InputsHandler::key_code_map
```

A map from key codes to corresponding string representations.

3.5.4.2 key_press_vec

```
std::vector<bool> InputsHandler::key_press_vec
```

A vector <bool> which indicates which keys are currently pressed. Useful for smooth movement.

3.5.4.3 key_pressed_once_vec

```
std::vector<bool> InputsHandler::key_pressed_once_vec
```

A vector (bool) which indicates which keys have been pressed once. Useful for discrete inputs.

3.5.4.4 mouse_left_click

```
bool InputsHandler::mouse_left_click
```

A boolean which indicates if the mouse left button has been clicked.

3.5.4.5 mouse_right_click

```
bool InputsHandler::mouse_right_click
```

A boolean which indicates if the mouse right button has been clicked.

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

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

3.6 MessagesHandler Class Reference

A class which handles message traffic between game objects.

```
#include <MessagesHandler.h>
```

Public Member Functions

- [MessagesHandler](#) (void)
Constructor for the [MessagesHandler](#) class.
- void [process](#) (void)
Method to process messages. To be called once per frame.
- [~MessagesHandler](#) (void)
Destructor for the [MessagesHandler](#) class.

3.6.1 Detailed Description

A class which handles message traffic between game objects.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 MessagesHandler()

```
MessagesHandler::MessagesHandler (
    void )
```

Constructor for the [MessagesHandler](#) class.

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

3.6.2.2 ~MessagesHandler()

```
MessagesHandler::~~MessagesHandler (
    void )
```

Destructor for the [MessagesHandler](#) class.

```
86 {
87     std::cout << "MessagesHandler at " << this << " destroyed" << std::endl;
88
89     return;
90 } /* ~MessagesHandler() */
```

3.6.3 Member Function Documentation

3.6.3.1 process()

```
void MessagesHandler::process (
    void )
```

Method to process messages. To be called once per frame.

```
67 {
68     //...
69
70     return;
71 } /* process() */
```

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

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

Variables

- const double `FLOAT_TOLERANCE` = 1e-6
Tolerance for floating point equality tests.
- 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`
- const std::vector< double > `tile_resource_cumulative_probabilities`

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 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 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

4.3.3.2 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

4.3.3.3 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

4.3.3.4 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

4.3.3.5 SECONDS_PER_FRAME

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

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

4.3.3.6 tile_resource_cumulative_probabilities

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

Initial value:

```
= {  
    0.10,  
    0.30,  
    0.70,  
    0.90,  
    1.00  
}
```

4.3.3.7 tile_type_cumulative_probabilities

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

Initial value:

```
= {  
    0.25,  
    0.50,  
    0.75,  
    1.00  
}
```

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.

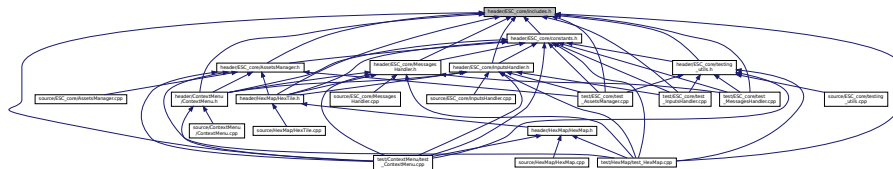
```
#include <chrono>  
#include <cmath>  
#include <cstdlib>  
#include <filesystem>  
#include <fstream>  
#include <iomanip>  
#include <iostream>  
#include <limits>  
#include <list>  
#include <map>  
#include <stdexcept>  
#include <sstream>  
#include <string>  
#include <vector>  
#include <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>
```

```
#include <SFML/Window.hpp>
```

Include dependency graph for includes.h:



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



4.5.1 Detailed Description

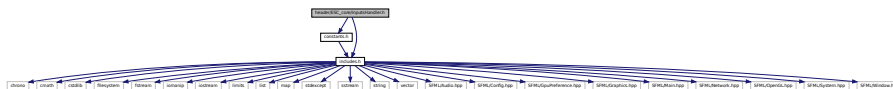
Header file for various includes.

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

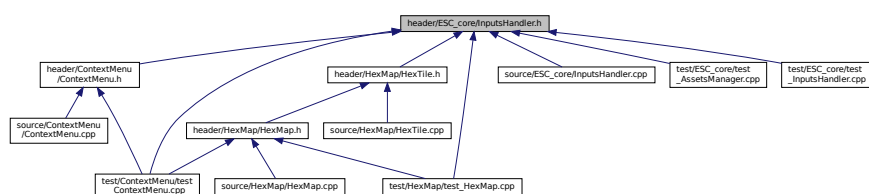
4.6 header/ESC_core/InputsHandler.h File Reference

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

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



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



Classes

- class [InputsHandler](#)

A class which handles inputs from peripherals (i.e., keyboard and mouse).

4.6.1 Detailed Description

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

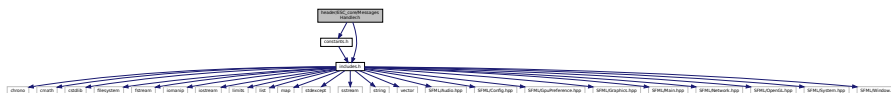
4.7 header/ESC_core/MessagesHandler.h File Reference

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

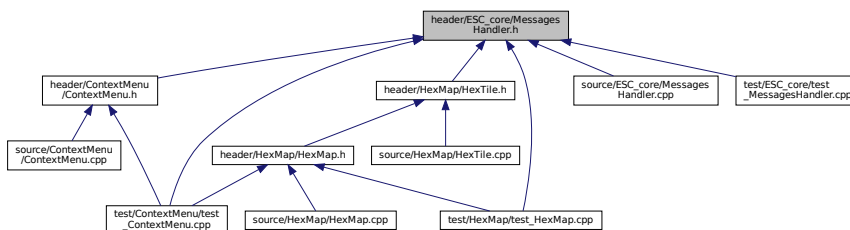
```
#include "constants.h"
```

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

Include dependency graph for MessagesHandler.h:



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



Classes

- class [MessagesHandler](#)

A class which handles message traffic between game objects.

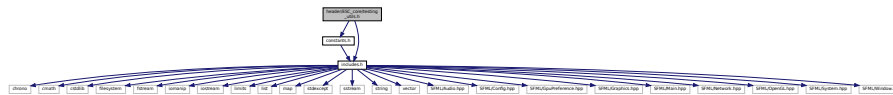
4.7.1 Detailed Description

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

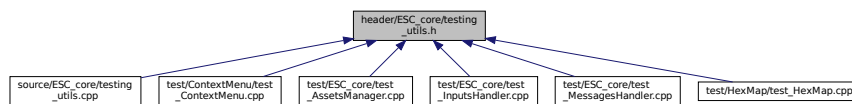
4.8 header/ESC_core/testing_utils.h File Reference

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.

4.8.1 Detailed Description

Header file for various testing utilities.

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

4.8.2 Function Documentation

4.8.2.1 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() */
```

4.8.2.2 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() */
```

4.8.2.3 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() */

```

4.8.2.4 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() */

```

4.8.2.5 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() */

```

4.8.2.6 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() */

```

4.8.2.7 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() */

```

4.8.2.8 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() */

```

4.8.2.9 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() */

```

4.8.2.10 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() */

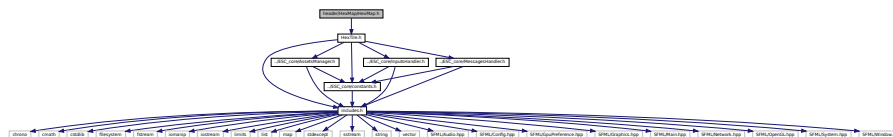
```

4.9 header/HexMap/HexMap.h File Reference

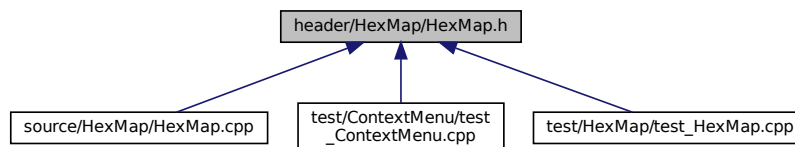
Header file for the [HexMap](#) class.

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

Include dependency graph for HexMap.h:



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



Classes

- class [HexMap](#)

A class which defines a hex map of hex tiles.

4.9.1 Detailed Description

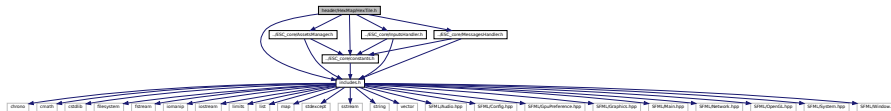
Header file for the [HexMap](#) class.

4.10 header/HexMap/HexTile.h File Reference

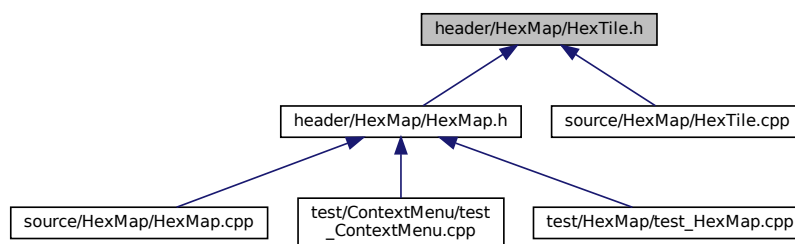
Header file for the [HexTile](#) class.

```
#include "../ESC_core/constants.h"
#include "../ESC_core/includes.h"
#include "../ESC_core/AssetsManager.h"
#include "../ESC_core/InputsHandler.h"
#include "../ESC_core/MessagesHandler.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](#) {
FOREST , LAKE , MOUNTAINS , OCEAN ,
PLAINS , N_TILE_TYPES }
- enum [TileResource](#) {
POOR , BELOW_AVERAGE , AVERAGE , ABOVE_AVERAGE ,
GOOD , N_TILE_RESOURCES }

An enumeration of the different tile types.

An enumeration of the different tile resource values.

4.10.1 Detailed Description

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.

```
51     {
52     POOR,
53     BELOW_AVERAGE,
54     AVERAGE,
55     ABOVE_AVERAGE,
56     GOOD,
57     N_TILE_RESOURCES
58 };
```

4.10.2.2 TileType

enum [TileType](#)

An enumeration of the different tile types.

Enumerator

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.

```
35     {
36     FOREST,
37     LAKE,
38     MOUNTAINS,
39     OCEAN,
40     PLAINS,
41     N_TILE_TYPES
42 };
```


4.13.1 Detailed Description

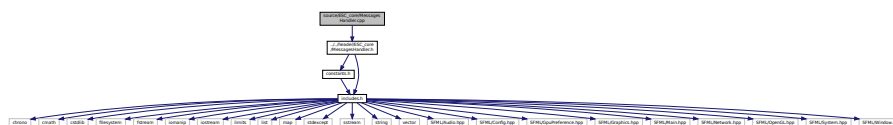
Implementation file for the `InputsHandler` class.

A class which handles inputs from peripherals (i.e., keyboard and mouse).

4.14 source/ESC_core/MessagesHandler.cpp File Reference

Implementation file for the `MessagesHandler` class.

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



4.14.1 Detailed Description

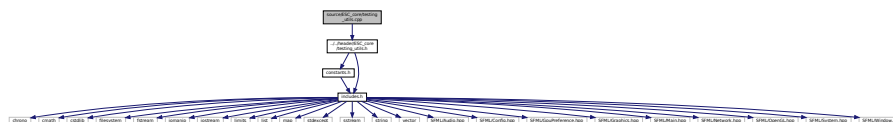
Implementation file for the `MessagesHandler` class.

A class which handles message traffic between game objects.

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 <code>std::cout</code> .
------------------	---

```

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 " <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() */

```

4.15.2.6 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__").

```

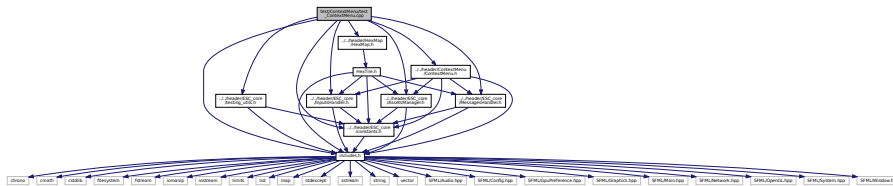
390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\n";
400     error_str += "Given statement is not 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/HexMap/HexMap.cpp File Reference

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


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



Functions

- int **main** (int argc, char **argv)

4.18.1 Detailed Description

Suite of tests for the [ContextMenu](#) class.

A suite of tests for the [ContextMenu](#) class.

4.18.2 Function Documentation

4.18.2.1 main()

```

int main (
    int argc,
    char ** argv )

42 {
43     #ifdef _WIN32
44         activateVirtualTerminal();
45     #endif /* _WIN32 */
46
47     printGold("\tTesting ContextMenu");
48     std::cout << std::endl;
49
50     srand(time(NULL));
51     int n_dots = 8;
52
53
54     try {
55         // 1. construct, load/open some test assets
56         AssetsManager assets_manager;
57         InputsHandler inputs_handler;
58         MessagesHandler messages_handler;
59
60         assets_manager.loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
61         assets_manager.loadFont("assets/fonts/Glass_TTY_VT220.ttf", "Glass_TTY_VT220");
62
63
64         // 2. test game loop
65         sf::Clock clock;
66         sf::Event event;
67         sf::RenderWindow window(
68             sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
69             "Testing ContextMenu"
70         );
71
72         double screen_width = window.getSize().x;

```

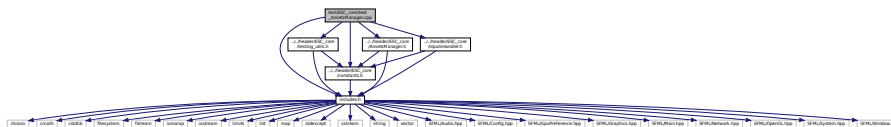
```

73     double screen_height = window.getSize().y;
74
75     testFloatEquals(
76         screen_width,
77         1200,
78         __FILE__,
79         __LINE__
80     );
81
82     testFloatEquals(
83         screen_height,
84         800,
85         __FILE__,
86         __LINE__
87     );
88
89     unsigned long long int frame = 0;
90     double time_since_run_s = 0;
91
92     ContextMenu context_menu(
93         &assets_manager,
94         &inputs_handler,
95         &messages_handler,
96         &window
97     );
98
99     HexMap hex_map(
100         6,
101         &assets_manager,
102         &inputs_handler,
103         &messages_handler,
104         &window
105     );
106
107     while (window.isOpen()) {
108         time_since_run_s = clock.getElapsedTime().asSeconds();
109
110         if (
111             time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
112         ) {
113             while (window.pollEvent(event))
114             {
115                 inputs_handler.process(&event);
116
117                 if (event.type == sf::Event::Closed) {
118                     window.close();
119                 }
120             }
121
122             context_menu.process();
123
124             //...
125
126             hex_map.process();
127
128             if (inputs_handler.key_pressed_once_vec[sf::Keyboard::Q]) {
129                 std::cout << "Q" << std::endl;
130                 hex_map.reroll();
131             }
132
133             if (inputs_handler.key_pressed_once_vec[sf::Keyboard::R]) {
134                 std::cout << "R" << std::endl;
135                 hex_map.toggleResourceOverlay();
136             }
137
138             if (inputs_handler.key_pressed_once_vec[sf::Keyboard::A]) {
139                 std::cout << "A" << std::endl;
140                 hex_map.assess();
141             }
142
143             window.clear();
144
145             hex_map.draw(); // draw hex map before context menu!
146             context_menu.draw();
147
148             window.display();
149
150             inputs_handler.reset();
151
152             std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
153             frame++;
154         }
155     }
156 }
157
158
159 catch (...) {

```


4.19 test/ESC_core/test_AssetsManager.cpp File Reference

```
#include ".../.../header/ESC_core/constants.h"
#include ".../.../header/ESC_core/includes.h"
#include ".../.../header/ESC_core/testing_utils.h"
#include ".../.../header/ESC_core/AssetsManager.h"
#include ".../.../header/ESC_core/InputsHandler.h"
Include dependency graph for test AssetsManager.cpp:
```



Functions

- int **main** (int argc, char **argv)

4.19.1 Detailed Description

Suite of tests for the [AssetsManager](#) class.

A suite of tests for the `AssetsManager` class.

4.19.2 Function Documentation

4.19.2.1 main()

```

int main (
    int argc,
    char ** argv )

38 {
39     #ifdef _WIN32
40         activateVirtualTerminal();
41     #endif /* _WIN32 */
42
43     printGold("\tTesting AssetsManager");
44     std::cout << std::endl;
45
46     srand(time(NULL));
47     int n_dots = 8;
48
49
50     try {
51         // 1. construct
52         InputsHandler inputs_handler;
53         AssetsManager assets_manager;
54
55
56         // 2. load/open some test assets
57         assets_manager.loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
58         assets_manager.loadTexture(
59             "assets/ESC_brand/ESC_key_98x81.png",
60             "ESC_key_98x81"
61         );
62         assets_manager.loadSound("assets/ESC_brand/key_press.ogg", "key_press");
63         assets_manager.loadTrack(
64             "assets/audio/tracks/AlexanderBlu_BackgroundElectronicModernMusic.ogg",
65             "AlexanderBlu_BackgroundElectronicModernMusic"
66         );
67
68
69         // 3. test game loop
70         sf::Clock clock;
71         sf::Event event;
72         sf::RenderWindow window(sf::VideoMode(800, 600), "Testing AssetsManager");
73
74         double screen_width = window.getSize().x;
75         double screen_height = window.getSize().y;
76
77         testFloatEquals(
78             screen_width,
79             800,
80             __FILE__,
81             __LINE__
82         );
83
84         testFloatEquals(
85             screen_height,
86             600,
87             __FILE__,
88             __LINE__
89         );
90
91         unsigned long long int frame = 0;
92         double time_since_run_s = 0;
93
94         assets_manager.playTrack();
95
96         sf::Sprite ESC_key(*(assets_manager.getTexture("ESC_key_98x81")));
97
98         double sprite_width = ESC_key.getLocalBounds().width;
99         double sprite_height = ESC_key.getLocalBounds().height;
100
101         double sprite_velocity_x = 256 * (2 * ((double)rand() / RAND_MAX) - 1);
102         double sprite_velocity_y = 256 * (2 * ((double)rand() / RAND_MAX) - 1);
103
104         ESC_key.setOrigin(sprite_width / 2, sprite_height / 2);
105         ESC_key.setPosition(
106             (screen_width - sprite_width) * ((double)rand() / RAND_MAX) + sprite_width / 2,
107             (screen_height - sprite_height) * ((double)rand() / RAND_MAX) + sprite_height / 2
108         );
109
110         sf::Text click_text(
111             "CLICK!",
112             *(assets_manager.getFont("DroidSansMono")),
113             16
114         );
115
116         double text_width = click_text.getLocalBounds().width;
117         double text_height = click_text.getLocalBounds().height;

```

```

118
119     click_text.setOrigin(text_width / 2, text_height / 2);
120
121     int alpha = 255;
122
123     click_text.setFillColor(sf::Color(255, 255, 255, alpha));
124
125     while (window.isOpen()) {
126         time_since_run_s = clock.getElapsedTime().asSeconds();
127
128         if (
129             time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
130         ) {
131             while (window.pollEvent(event))
132             {
133                 //...
134
135                 if (event.type == sf::Event::Closed) {
136                     window.close();
137                 }
138             }
139
140             ESC_key.move(
141                 sprite_velocity_x * SECONDS_PER_FRAME,
142                 sprite_velocity_y * SECONDS_PER_FRAME
143             );
144
145             if (
146                 ESC_key.getPosition().x <= sprite_width / 2 or
147                 ESC_key.getPosition().x >= screen_width - sprite_width / 2
148             ) {
149                 sprite_velocity_x *= -1;
150
151                 assets_manager.getSound("key_press")->play();
152
153                 alpha = 255;
154                 click_text.setPosition(
155                     ESC_key.getPosition().x,
156                     ESC_key.getPosition().y
157                 );
158             }
159
160             if (
161                 ESC_key.getPosition().y <= sprite_height / 2 or
162                 ESC_key.getPosition().y >= screen_height - sprite_height / 2
163             ) {
164                 sprite_velocity_y *= -1;
165
166                 assets_manager.getSound("key_press")->play();
167
168                 alpha = 255;
169                 click_text.setPosition(
170                     ESC_key.getPosition().x,
171                     ESC_key.getPosition().y
172                 );
173             }
174
175             window.clear();
176
177             window.draw(ESC_key);
178             window.draw(click_text);
179
180             window.display();
181
182             alpha -= 8;
183             if (alpha < 0) {
184                 alpha = 0;
185             }
186
187             click_text.setFillColor(sf::Color(255, 255, 255, alpha));
188
189             std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
190             frame++;
191         }
192     }
193 }
194
195
196 catch (...) {
197     //...
198
199     printGold(" ");
200     for (int i = 0; i < n_dots; i++) {
201         printGold(".");
202     }
203     printGold(" ");
204     printRed("FAIL");

```


4.20.2.1 main()

```

int main (
    int argc,
    char ** argv )
37 {
38     #ifdef _WIN32
39         activateVirtualTerminal();
40     #endif /* _WIN32 */
41
42     printGold("\tTesting InputsHandler");
43     std::cout << std::endl;
44
45     srand(time(NULL));
46     int n_dots = 8;
47
48
49     try {
50         // 1. construct and spot check attributes
51         InputsHandler inputs_handler;
52
53         testFloatEquals(
54             int(sf::Keyboard::KeyCount),
55             101,
56             __FILE__,
57             __LINE__
58         );
59
60         testFloatEquals(
61             inputs_handler.key_press_vec.size(),
62             int(sf::Keyboard::KeyCount),
63             __FILE__,
64             __LINE__
65         );
66
67         testFloatEquals(
68             inputs_handler.key_pressed_once_vec.size(),
69             int(sf::Keyboard::KeyCount),
70             __FILE__,
71             __LINE__
72         );
73
74
75         // 2. test game loop
76         sf::Clock clock;
77         sf::Event event;
78         sf::RenderWindow window(sf::VideoMode(800, 600), "Testing InputsHandler");
79
80         double screen_width = window.getSize().x;
81         double screen_height = window.getSize().y;
82
83         testFloatEquals(
84             screen_width,
85             800,
86             __FILE__,
87             __LINE__
88         );
89
90         testFloatEquals(
91             screen_height,
92             600,
93             __FILE__,
94             __LINE__
95         );
96
97         unsigned long long int frame = 0;
98         double time_since_run_s = 0;
99
100         while (window.isOpen()) {
101             time_since_run_s = clock.getElapsedTime().asSeconds();
102
103             if (
104                 time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
105             ) {
106                 while (window.pollEvent(event))
107                 {
108                     inputs_handler.process(&event);
109
110                     if (event.type == sf::Event::Closed) {
111                         window.close();
112                     }
113                 }
114
115                 window.clear();
116                 window.display();

```

```

117
118         inputs_handler.printKeysPressed();
119
120         if (inputs_handler.key_pressed_once_vec[sf::Keyboard::Enter]) {
121             std::cout << "Enter" << std::endl;
122         }
123
124
125
126         inputs_handler.reset();
127
128         std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
129         frame++;
130     }
131 }
132 }
133
134
135 catch (...) {
136     //...
137
138     printGold(" ");
139     for (int i = 0; i < n_dots; i++) {
140         printGold(".");
141     }
142     printGold(" ");
143     printRed("FAIL");
144     std::cout << std::endl;
145     throw;
146 }
147
148
149 //...
150
151 printGold(" ");
152 for (int i = 0; i < n_dots; i++) {
153     printGold(".");
154 }
155 printGold(" ");
156 printGreen("PASS");
157 std::cout << std::endl;
158
159 return 0;
160 } /* main() */

```

4.21 test/ESC_core/test_MessagesHandler.cpp File Reference

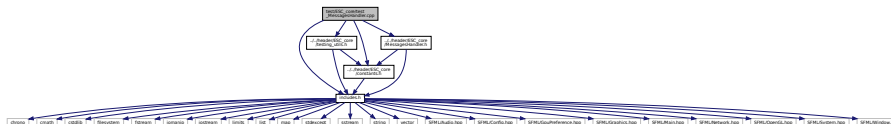
Suite of tests for the [MessagesHandler](#) class.

```

#include "../..header/ESC_core/constants.h"
#include "../..header/ESC_core/includes.h"
#include "../..header/ESC_core/testing_utils.h"
#include "../..header/ESC_core/MessagesHandler.h"

```

Include dependency graph for test_MessagesHandler.cpp:



Functions

- int [main](#) (int argc, char **argv)

4.21.1 Detailed Description

Suite of tests for the [MessagesHandler](#) class.

A suite of tests for the [MessagesHandler](#) class.

4.21.2 Function Documentation

4.21.2.1 main()

```

int main (
    int argc,
    char ** argv )
37 {
38     #ifdef _WIN32
39         activateVirtualTerminal();
40     #endif /* _WIN32 */
41
42     printGold("\tTesting MessagesHandler");
43     std::cout << std::endl;
44
45     srand(time(NULL));
46     int n_dots = 8;
47
48
49     try {
50         // 1. construct
51         MessagesHandler messages_handler;
52
53
54         // 2. test game loop
55         sf::Clock clock;
56         sf::Event event;
57         sf::RenderWindow window(sf::VideoMode(800, 600), "Testing MessagesHandler");
58
59         double screen_width = window.getSize().x;
60         double screen_height = window.getSize().y;
61
62         testFloatEquals(
63             screen_width,
64             800,
65             __FILE__,
66             __LINE__
67         );
68
69         testFloatEquals(
70             screen_height,
71             600,
72             __FILE__,
73             __LINE__
74         );
75
76         unsigned long long int frame = 0;
77         double time_since_run_s = 0;
78
79         while (window.isOpen()) {
80             time_since_run_s = clock.getElapsedTime().asSeconds();
81
82             if (
83                 time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
84             ) {
85                 while (window.pollEvent(event))
86                 {
87                     //...
88
89                     if (event.type == sf::Event::Closed) {
90                         window.close();
91                     }
92                 }
93
94                 window.clear();
95                 window.display();
96
97                 std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
98                 frame++;
99             }
100         }
101     }
102
103
104     catch (...) {
105         //...
106
107         printGold(" ");

```

```

108         for (int i = 0; i < n_dots; i++) {
109             printGold(".");
110         }
111         printGold(" ");
112         printRed("FAIL");
113         std::cout << std::endl;
114         throw;
115     }
116
117
118     //...
119
120     printGold(" ");
121     for (int i = 0; i < n_dots; i++) {
122         printGold(".");
123     }
124     printGold(" ");
125     printGreen("PASS");
126     std::cout << std::endl;
127
128     return 0;
129 } /* main() */

```

4.22 test/HexMap/test_HexMap.cpp File Reference

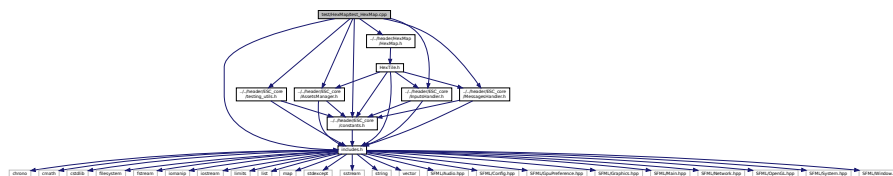
Suite of tests for the [HexMap](#) class.

```

#include "../..header/ESC_core/constants.h"
#include "../..header/ESC_core/includes.h"
#include "../..header/ESC_core/testing_utils.h"
#include "../..header/ESC_core/AssetsManager.h"
#include "../..header/ESC_core/InputsHandler.h"
#include "../..header/ESC_core/MessagesHandler.h"
#include "../..header/HexMap/HexMap.h"

```

Include dependency graph for test_HexMap.cpp:



Functions

- [int main](#) (int argc, char **argv)

4.22.1 Detailed Description

Suite of tests for the [HexMap](#) class.

A suite of tests for the [HexMap](#) class.

4.22.2 Function Documentation

4.22.2.1 main()

```

int main (
    int argc,
    char ** argv )
{
    41 {
    42     #ifdef _WIN32
    43         activateVirtualTerminal();
    44     #endif /* _WIN32 */
    45
    46     printGold("\tTesting HexMap");
    47     std::cout << std::endl;
    48
    49     srand(time(NULL));
    50     int n_dots = 8;
    51
    52
    53     try {
    54         // 1. construct, load/open some test assets
    55         AssetsManager assets_manager;
    56         InputsHandler inputs_handler;
    57         MessagesHandler messages_handler;
    58
    59         assets_manager.loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
    60
    61
    62         // 2. test game loop
    63         sf::Clock clock;
    64         sf::Event event;
    65         sf::RenderWindow window(
    66             sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
    67             "Testing HexMap"
    68         );
    69
    70         double screen_width = window.getSize().x;
    71         double screen_height = window.getSize().y;
    72
    73         testFloatEquals(
    74             screen_width,
    75             1200,
    76             __FILE__,
    77             __LINE__
    78         );
    79
    80         testFloatEquals(
    81             screen_height,
    82             800,
    83             __FILE__,
    84             __LINE__
    85         );
    86
    87         unsigned long long int frame = 0;
    88         double time_since_run_s = 0;
    89
    90         HexMap hex_map(
    91             6,
    92             &assets_manager,
    93             &inputs_handler,
    94             &messages_handler,
    95             &window
    96         );
    97
    98         while (window.isOpen()) {
    99             time_since_run_s = clock.getElapsedTime().asSeconds();
    100
    101             if (
    102                 time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
    103             ) {
    104                 while (window.pollEvent(event))
    105                 {
    106                     inputs_handler.process(&event);
    107
    108                     if (event.type == sf::Event::Closed) {
    109                         window.close();
    110                     }
    111                 }
    112
    113                 hex_map.process();
    114
    115                 if (inputs_handler.key_pressed_once_vec[sf::Keyboard::Q]) {
    116                     std::cout << "Q" << std::endl;
    117                     hex_map.reroll();
    118                 }
    119
    120                 if (inputs_handler.key_pressed_once_vec[sf::Keyboard::R]) {

```

```
121             std::cout << "R" << std::endl;
122             hex_map.toggleResourceOverlay();
123         }
124
125         if (inputs_handler.key_pressed_once_vec[sf::Keyboard::A]) {
126             std::cout << "A" << std::endl;
127             hex_map.assess();
128         }
129
130         window.clear();
131
132         hex_map.draw();
133
134         window.display();
135
136         inputs_handler.reset();
137
138         std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
139         frame++;
140     }
141 }
142 }
143
144
145 catch (...) {
146     //...
147
148     printGold(" ");
149     for (int i = 0; i < n_dots; i++) {
150         printGold(".");
151     }
152     printGold(" ");
153     printRed("FAIL");
154     std::cout << std::endl;
155     throw;
156 }
157
158
159 //...
160
161 printGold(" ");
162 for (int i = 0; i < n_dots; i++) {
163     printGold(".");
164 }
165 printGold(" ");
166 printGreen("PASS");
167 std::cout << std::endl;
168
169 return 0;
170 } /* main() */
```

Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL <https://www.sfml-dev.org/>. 81

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 80

Wikipedia. Hexagon, 2023. URL <https://en.wikipedia.org/wiki/Hexagon>. 51

Index

- __assembleHexMap
 - HexMap, [34](#)
- __constructKeyCodeMap
 - InputsHandler, [64](#)
- __drawConsoleScreenFrame
 - ContextMenu, [20](#)
- __drawConsoleText
 - ContextMenu, [20](#)
- __drawVisualScreenFrame
 - ContextMenu, [21](#)
- __enforceOceanContinuity
 - HexMap, [34](#)
- __getMajorityTileType
 - HexMap, [35](#)
- __getNeighboursVector
 - HexMap, [36](#)
- __getNoise
 - HexMap, [36](#)
- __getSelectedTile
 - HexMap, [38](#)
- __getValidMapIndexPositions
 - HexMap, [38](#)
- __isClicked
 - HexTile, [52](#)
- __isLakeTouchingOcean
 - HexMap, [39](#)
- __layTiles
 - HexMap, [40](#)
- __loadSoundBuffer
 - AssetsManager, [7](#)
- __procedurallyGenerateTileResources
 - HexMap, [42](#)
- __procedurallyGenerateTileTypes
 - HexMap, [42](#)
- __setResourceText
 - HexTile, [52](#)
- __setUpConsoleScreen
 - ContextMenu, [21](#)
- __setUpConsoleScreenFrame
 - ContextMenu, [22](#)
- __setUpGlassScreen
 - HexMap, [43](#)
- __setUpMenuFrame
 - ContextMenu, [24](#)
- __setUpNodeSprite
 - HexTile, [53](#)
- __setUpResourceChipSprite
 - HexTile, [54](#)
- __setUpSelectOutlineSprite
 - HexTile, [54](#)
- __setUpTileSprite
 - HexTile, [54](#)
- __setUpVisualScreen
 - ContextMenu, [24](#)
- __setUpVisualScreenFrame
 - ContextMenu, [24](#)
- __smoothTileTypes
 - HexMap, [43](#)
- ~AssetsManager
 - AssetsManager, [6](#)
- ~ContextMenu
 - ContextMenu, [20](#)
- ~HexMap
 - HexMap, [33](#)
- ~HexTile
 - HexTile, [52](#)
- ~InputsHandler
 - InputsHandler, [63](#)
- ~MessagesHandler
 - MessagesHandler, [71](#)
- ABOVE_AVERAGE
 - HexTile.h, [91](#)
- assess
 - HexMap, [44](#)
 - HexTile, [55](#)
- assets_manager_ptr
 - ContextMenu, [27](#)
 - HexMap, [46](#)
 - HexTile, [59](#)
- 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, 91
- BELOW_AVERAGE
 - HexTile.h, 91
- border_tiles_vec
 - HexMap, 46
- clear
 - AssetsManager, 8
 - HexMap, 44
- console_message
 - ContextMenu, 27
- console_screen
 - ContextMenu, 27
- console_screen_frame_bottom
 - ContextMenu, 27
- console_screen_frame_left
 - ContextMenu, 28
- console_screen_frame_right
 - ContextMenu, 28
- console_screen_frame_top
 - ContextMenu, 28
- constants.h
 - FLOAT_TOLERANCE, 78
 - FOREST_GREEN, 76
 - FRAMES_PER_SECOND, 79
 - GAME_HEIGHT, 79
 - GAME_WIDTH, 79
 - LAKE_BLUE, 76
 - MENU_FRAME_GREY, 76
 - MONOCHROME_SCREEN_BACKGROUND, 77
 - MONOCHROME_TEXT_AMBER, 77
 - MONOCHROME_TEXT_GREEN, 77
 - MONOCHROME_TEXT_RED, 77
 - MOUNTAINS_GREY, 77
 - OCEAN_BLUE, 78
 - PLAINS_YELLOW, 78
 - SECONDS_PER_FRAME, 79
 - tile_resource_cumulative_probabilities, 79
 - tile_type_cumulative_probabilities, 79
 - VISUAL_SCREEN_FRAME_GREY, 78
- ContextMenu, 17
 - __drawConsoleScreenFrame, 20
 - __drawConsoleText, 20
 - __drawVisualScreenFrame, 21
 - __setUpConsoleScreen, 21
 - __setUpConsoleScreenFrame, 22
 - __setUpMenuFrame, 24
 - __setUpVisualScreen, 24
 - __setUpVisualScreenFrame, 24
 - ~ContextMenu, 20
- assets_manager_ptr, 27
- console_message, 27
- console_screen, 27
- console_screen_frame_bottom, 27
- console_screen_frame_left, 28
- console_screen_frame_right, 28
- console_screen_frame_top, 28
- ContextMenu, 19
- draw, 26
- frame, 28
- game_menu_up, 28
- inputs_handler_ptr, 28
- menu_frame, 29
- messages_handler_ptr, 29
- position_x, 29
- position_y, 29
- process, 26
- render_window_ptr, 29
- visual_screen, 29
- visual_screen_frame_bottom, 30
- visual_screen_frame_left, 30
- visual_screen_frame_right, 30
- visual_screen_frame_top, 30
- current_track
 - AssetsManager, 16
- draw
 - ContextMenu, 26
 - HexMap, 44
 - HexTile, 55
- expectedErrorNotDetected
 - testing_utils.cpp, 94
 - testing_utils.h, 84
- FLOAT_TOLERANCE
 - constants.h, 78
- font_map
 - AssetsManager, 16
- FOREST
 - HexTile.h, 91
- FOREST_GREEN
 - constants.h, 76
- frame
 - ContextMenu, 28
 - HexMap, 46
 - HexTile, 59
- FRAMES_PER_SECOND
 - constants.h, 79
- GAME_HEIGHT
 - constants.h, 79
- game_menu_up
 - ContextMenu, 28
- GAME_WIDTH
 - constants.h, 79
- getCurrentTrackKey
 - AssetsManager, 9
- getFont

- AssetsManager, 9
- getSound
 - AssetsManager, 10
- getSoundBuffer
 - AssetsManager, 10
- getTexture
 - AssetsManager, 11
- getTrackStatus
 - AssetsManager, 11
- glass_screen
 - HexMap, 47
- GOOD
 - HexTile.h, 91
- header/ContextMenu/ContextMenu.h, 73
- header/ESC_core/AssetsManager.h, 74
- header/ESC_core/constants.h, 75
- header/ESC_core/doxygen_cite.h, 80
- header/ESC_core/includes.h, 80
- header/ESC_core/InputsHandler.h, 81
- header/ESC_core/MessagesHandler.h, 82
- header/ESC_core/testing_utils.h, 83
- header/HexMap/HexMap.h, 89
- header/HexMap/HexTile.h, 90
- hex_map
 - HexMap, 47
- HexMap, 31
 - __assembleHexMap, 34
 - __enforceOceanContinuity, 34
 - __getMajorityTileType, 35
 - __getNeighboursVector, 36
 - __getNoise, 36
 - __getSelectedTile, 38
 - __getValidMapIndexPositions, 38
 - __isLakeTouchingOcean, 39
 - __layTiles, 40
 - __procedurallyGenerateTileResources, 42
 - __procedurallyGenerateTileTypes, 42
 - __setUpGlassScreen, 43
 - __smoothTileTypes, 43
 - ~HexMap, 33
 - assess, 44
 - assets_manager_ptr, 46
 - border_tiles_vec, 46
 - clear, 44
 - draw, 44
 - frame, 46
 - glass_screen, 47
 - hex_map, 47
 - HexMap, 33
 - inputs_handler_ptr, 47
 - messages_handler_ptr, 47
 - n_layers, 47
 - n_tiles, 47
 - position_x, 48
 - position_y, 48
 - process, 45
 - render_window_ptr, 48
 - reroll, 45
 - tile_position_x_vec, 48
 - tile_position_y_vec, 48
 - toggleResourceOverlay, 46
- HexTile, 49
 - __isClicked, 52
 - __setResourceText, 52
 - __setUpNodeSprite, 53
 - __setUpResourceChipSprite, 54
 - __setUpSelectOutlineSprite, 54
 - __setUpTileSprite, 54
 - ~HexTile, 52
 - assess, 55
 - assets_manager_ptr, 59
 - draw, 55
 - frame, 59
 - HexTile, 51
 - inputs_handler_ptr, 59
 - is_selected, 59
 - major_radius, 60
 - messages_handler_ptr, 60
 - minor_radius, 60
 - node_sprite, 60
 - position_x, 60
 - position_y, 60
 - process, 56
 - render_window_ptr, 61
 - resource_assessed, 61
 - resource_chip_sprite, 61
 - resource_text, 61
 - select_outline_sprite, 61
 - setTileResource, 56, 57
 - setTileType, 57, 58
 - show_node, 61
 - show_resource, 62
 - tile_resource, 62
 - tile_sprite, 62
 - tile_type, 62
 - toggleResourceOverlay, 59
- HexTile.h
 - ABOVE_AVERAGE, 91
 - AVERAGE, 91
 - BELOW_AVERAGE, 91
 - FOREST, 91
 - GOOD, 91
 - LAKE, 91
 - MOUNTAINS, 91
 - N_TILE_RESOURCES, 91
 - N_TILE_TYPES, 91
 - OCEAN, 91
 - PLAINS, 91
 - POOR, 91
 - TileResource, 91
 - TileType, 91
- inputs_handler_ptr
 - ContextMenu, 28
 - HexMap, 47
 - HexTile, 59
- InputsHandler, 62

- __constructKeyCodeMap, 64
 - ~InputsHandler, 63
 - InputsHandler, 63
 - key_code_map, 69
 - key_press_vec, 69
 - key_pressed_once_vec, 69
 - mouse_left_click, 70
 - mouse_right_click, 70
 - printKeysPressed, 68
 - process, 68
 - reset, 69
- is_selected
 - HexTile, 59
- key_code_map
 - InputsHandler, 69
- key_press_vec
 - InputsHandler, 69
- key_pressed_once_vec
 - InputsHandler, 69
- LAKE
 - HexTile.h, 91
- LAKE_BLUE
 - constants.h, 76
- loadFont
 - AssetsManager, 12
- loadSound
 - AssetsManager, 12
- loadTexture
 - AssetsManager, 13
- loadTrack
 - AssetsManager, 14
- main
 - test_AssetsManager.cpp, 103
 - test_ContextMenu.cpp, 101
 - test_HexMap.cpp, 110
 - test_InputsHandler.cpp, 106
 - test_MessagesHandler.cpp, 109
- major_radius
 - HexTile, 60
- menu_frame
 - ContextMenu, 29
- MENU_FRAME_GREY
 - constants.h, 76
- messages_handler_ptr
 - ContextMenu, 29
 - HexMap, 47
 - HexTile, 60
- MessagesHandler, 70
 - ~MessagesHandler, 71
 - MessagesHandler, 71
 - process, 71
- minor_radius
 - HexTile, 60
- MONOCHROME_SCREEN_BACKGROUND
 - constants.h, 77
- MONOCHROME_TEXT_AMBER
 - constants.h, 77
- MONOCHROME_TEXT_GREEN
 - constants.h, 77
- MONOCHROME_TEXT_RED
 - constants.h, 77
- MOUNTAINS
 - HexTile.h, 91
- MOUNTAINS_GREY
 - constants.h, 77
- mouse_left_click
 - InputsHandler, 70
- mouse_right_click
 - InputsHandler, 70
- n_layers
 - HexMap, 47
- N_TILE_RESOURCES
 - HexTile.h, 91
- N_TILE_TYPES
 - HexTile.h, 91
- n_tiles
 - HexMap, 47
- nextTrack
 - AssetsManager, 14
- node_sprite
 - HexTile, 60
- OCEAN
 - HexTile.h, 91
- OCEAN_BLUE
 - constants.h, 78
- pauseTrack
 - AssetsManager, 15
- PLAINS
 - HexTile.h, 91
- PLAINS_YELLOW
 - constants.h, 78
- playTrack
 - AssetsManager, 15
- POOR
 - HexTile.h, 91
- position_x
 - ContextMenu, 29
 - HexMap, 48
 - HexTile, 60
- position_y
 - ContextMenu, 29
 - HexMap, 48
 - HexTile, 60
- previousTrack
 - AssetsManager, 15
- printGold
 - testing_utils.cpp, 95
 - testing_utils.h, 84
- printGreen
 - testing_utils.cpp, 95
 - testing_utils.h, 84
- printKeysPressed

- InputsHandler, 68
- printRed
 - testing_utils.cpp, 95
 - testing_utils.h, 85
- process
 - ContextMenu, 26
 - HexMap, 45
 - HexTile, 56
 - InputsHandler, 68
 - MessagesHandler, 71
- render_window_ptr
 - ContextMenu, 29
 - HexMap, 48
 - HexTile, 61
- reroll
 - HexMap, 45
- reset
 - InputsHandler, 69
- resource_assessed
 - HexTile, 61
- resource_chip_sprite
 - HexTile, 61
- resource_text
 - HexTile, 61
- SECONDS_PER_FRAME
 - constants.h, 79
- select_outline_sprite
 - HexTile, 61
- setTileResource
 - HexTile, 56, 57
- setTileType
 - HexTile, 57, 58
- show_node
 - HexTile, 61
- show_resource
 - HexTile, 62
- sound_map
 - AssetsManager, 16
- soundbuffer_map
 - AssetsManager, 16
- source/ContextMenu/ContextMenu.cpp, 92
- source/ESC_core/AssetsManager.cpp, 92
- source/ESC_core/InputsHandler.cpp, 92
- source/ESC_core/MessagesHandler.cpp, 93
- source/ESC_core/testing_utils.cpp, 93
- source/HexMap/HexMap.cpp, 99
- source/HexMap/HexTile.cpp, 100
- stopTrack
 - AssetsManager, 15
- test/ContextMenu/test_ContextMenu.cpp, 100
- test/ESC_core/test_AssetsManager.cpp, 103
- test/ESC_core/test_InputsHandler.cpp, 106
- test/ESC_core/test_MessagesHandler.cpp, 108
- test/HexMap/test_HexMap.cpp, 110
- test_AssetsManager.cpp
 - main, 103
- test_ContextMenu.cpp
 - main, 101
- test_HexMap.cpp
 - main, 110
- test_InputsHandler.cpp
 - main, 106
- test_MessagesHandler.cpp
 - main, 109
- testFloatEquals
 - testing_utils.cpp, 96
 - testing_utils.h, 85
- testGreaterThan
 - testing_utils.cpp, 96
 - testing_utils.h, 86
- testGreaterThanOrEqualTo
 - testing_utils.cpp, 97
 - testing_utils.h, 86
- testing_utils.cpp
 - expectedErrorNotDetected, 94
 - printGold, 95
 - printGreen, 95
 - printRed, 95
 - testFloatEquals, 96
 - testGreaterThan, 96
 - testGreaterThanOrEqualTo, 97
 - testLessThan, 98
 - testLessThanOrEqualTo, 98
 - testTruth, 99
- testing_utils.h
 - expectedErrorNotDetected, 84
 - printGold, 84
 - printGreen, 84
 - printRed, 85
 - testFloatEquals, 85
 - testGreaterThan, 86
 - testGreaterThanOrEqualTo, 86
 - testLessThan, 87
 - testLessThanOrEqualTo, 88
 - testTruth, 88
- testLessThan
 - testing_utils.cpp, 98
 - testing_utils.h, 87
- testLessThanOrEqualTo
 - testing_utils.cpp, 98
 - testing_utils.h, 88
- testTruth
 - testing_utils.cpp, 99
 - testing_utils.h, 88
- texture_map
 - AssetsManager, 16
- tile_position_x_vec
 - HexMap, 48
- tile_position_y_vec
 - HexMap, 48
- tile_resource
 - HexTile, 62
- tile_resource_cumulative_probabilities
 - constants.h, 79

- tile_sprite
 - HexTile, [62](#)
- tile_type
 - HexTile, [62](#)
- tile_type_cumulative_probabilities
 - constants.h, [79](#)
- TileResource
 - HexTile.h, [91](#)
- TileType
 - HexTile.h, [91](#)
- toggleResourceOverlay
 - HexMap, [46](#)
 - HexTile, [59](#)
- track_map
 - AssetsManager, [17](#)

- visual_screen
 - ContextMenu, [29](#)
- visual_screen_frame_bottom
 - ContextMenu, [30](#)
- VISUAL_SCREEN_FRAME_GREY
 - constants.h, [78](#)
- visual_screen_frame_left
 - ContextMenu, [30](#)
- visual_screen_frame_right
 - ContextMenu, [30](#)
- visual_screen_frame_top
 - ContextMenu, [30](#)