# Road To Zero

Generated by Doxygen 1.9.1

# Chapter 1

# Class Index

## 1.1  Class List

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

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# 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**

| path_2_sound | A path (either relative or absolute) to the sound file. |
|---|---|
| sound_key | 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
662      //  2. clear textures
663      std::map<std::string, sf::Texture*>::iterator texture_iter;
664      for (
665          texture_iter = this->texture_map.begin();
666          texture_iter != this->texture_map.end();
667          texture_iter++
668      ) {
669          delete texture_iter->second;
670
671          std::cout « "Texture " « texture_iter->first « " deleted from texture map" «
672              std::endl;
673      }
674      this->texture_map.clear();
675
676
677      //  3. clear sound buffers
678      std::map<std::string, sf::SoundBuffer*>::iterator soundbuffer_iter;
679      for (
680          soundbuffer_iter = this->soundbuffer_map.begin();
681          soundbuffer_iter != this->soundbuffer_map.end();
682          soundbuffer_iter++
683      ) {
684          delete soundbuffer_iter->second;
685
686          std::cout « "SoundBuffer " « soundbuffer_iter->first «
687              " deleted from soundbuffer map" « std::endl;
688      }
689      this->soundbuffer_map.clear();
690
691
692      //  4. clear sounds
693      std::map<std::string, sf::Sound*>::iterator sound_iter;
694      for (
695          sound_iter = this->sound_map.begin();
696          sound_iter != this->sound_map.end();
697          sound_iter++
698      ) {
699          sound_iter->second->stop();
700          delete sound_iter->second;
701
702          std::cout « "Sound " « sound_iter->first « " deleted from sound map" «
703              std::endl;
704      }
705      this->sound_map.clear();
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**

| | |
|---|---|
| *font_key* | 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**

| *sound_key* | 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**

| *sound_key* | 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**

| texture_key | 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**

| *path_2_font* | A path (either relative or absolute) to the font file. |
| *font_key* | 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
152     //  2. load from file, throw error on fail
153     sf::Font* font_ptr = new sf::Font();
154
155     if (not font_ptr->loadFromFile(path_2_font)) {
156         std::string error_str = "ERROR  AssetsManager::loadFont()  could not load ";
157         error_str += "font at ";
158         error_str += path_2_font;
159
160         this->clear();
161
162         #ifdef _WIN32
163             std::cout « error_str « std::endl;
164         #endif  /* _WIN32 */
165
166         throw std::runtime_error(error_str);
167     }
168
169
170     //  3. insert into font map
171     this->font_map.insert(std::pair<std::string, sf::Font*>(font_key, font_ptr));
172
173     std::cout « "Font " « font_key « " inserted into font map" « std::endl;
174
175     return;
176 }   /* 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**

| *path_2_sound* | A path (either relative or absolute) to the sound file. |
| --- | --- |
| *sound_key* | 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**

| *path_2_texture* | A path (either relative or absolute) to the texture file. |
| --- | --- |
| *texture_key* | 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
213     //  2. load from file, throw error on fail
214     sf::Texture* texture_ptr = new sf::Texture();
215
216     if (not texture_ptr->loadFromFile(path_2_texture)) {
217         std::string error_str = "ERROR  AssetsManager::loadTexture()  could not load ";
218         error_str += "texture at ";
219         error_str += path_2_texture;
220
221         this->clear();
222
223         #ifdef _WIN32
224             std::cout « error_str « std::endl;
```

```
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**

| | |
|---|---|
| *path_2_track* | A path (either relative or absolute) to the track file. |
| *track_key* | A key associated with the track (for indexing into the track map). |

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

### 3.1.3.13 nextTrack()

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

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

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

### 3.1.3.14 pauseTrack()

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

Method to pause the current track.

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

### 3.1.3.15 playTrack()

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

Method to play the current track.

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

### 3.1.3.16 previousTrack()

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

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

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

**3.1.3.17 stopTrack()**

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

Method to stop the current track.

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

**3.1.4 Member Data Documentation**

**3.1.4.1 current_track**

```
std::map<std::string, sf::Music*>::iterator AssetsManager::current_track
```

A map iterator which corresponds to the current track (i.e., the track currently being played).

**3.1.4.2 font_map**

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

A map of pointers to loaded fonts.

**3.1.4.3 sound_map**

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

A map of pointers to loaded sounds.

**3.1.4.4 soundbuffer_map**

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

A map of pointers to sound buffers.

**3.1.4.5 texture_map**

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

A map of pointers to loaded textures.

**3.1.4.6 track_map**

`std::map<std::string, sf::Music*> AssetsManager::track_map`

A map of pointers to opened tracks (i.e. sf::Music).

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

- header/ESC_core/AssetsManager.h
- source/ESC_core/AssetsManager.cpp

## 3.2 ContextMenu Class Reference

A class which defines a context menu for the game.

`#include <ContextMenu.h>`

Collaboration diagram for ContextMenu:
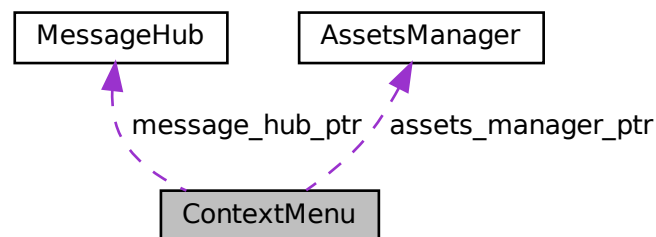


### Public Member Functions

- ContextMenu (sf::Event *, sf::RenderWindow *, AssetsManager *, MessageHub *)

    *Constructor for the ContextMenu class.*
- void processEvent (void)

    *Method to processEvent ContextMenu. To be called once per event.*
- void processMessage (void)

    *Method to processMessage ContextMenu. To be called once per message.*
- void draw (void)

    *Method to draw the hex tile to the render window. To be called once per frame.*
- ∼ContextMenu (void)

    *Destructor for the ContextMenu class.*

## Public Attributes

- ConsoleState console_state

    *The current state of the console screen.*
- bool game_menu_up

    *Indicates whether or not the game menu is up.*
- int frame

    *The current frame of this object.*
- double position_x

    *The position of the object.*
- double position_y

    *The position of the object.*
- std::string console_string

    *The string to be printed to the console screen.*
- sf::RectangleShape menu_frame

    *The frame of the context menu.*
- sf::RectangleShape visual_screen

    *The context menu screen for visuals.*
- sf::ConvexShape visual_screen_frame_top

    *The top framing of the visual screen.*
- sf::ConvexShape visual_screen_frame_left

    *The left framing of the visual screen.*
- sf::ConvexShape visual_screen_frame_bottom

    *The bottom framing of the visual screen.*
- sf::ConvexShape visual_screen_frame_right

    *The right framing of the visual screen.*
- sf::RectangleShape console_screen

    *The context menu console screen (for animated text output).*
- sf::ConvexShape console_screen_frame_top

    *The top framing of the console screen.*
- sf::ConvexShape console_screen_frame_left

    *The left framing of the console screen.*
- sf::ConvexShape console_screen_frame_bottom

    *The bottom framing of the console screen.*
- sf::ConvexShape console_screen_frame_right

    *The right framing of the console screen.*

## Private Member Functions

- void __setUpMenuFrame (void)

    *Helper method to set up context menu frame (drawable).*
- void __setUpVisualScreen (void)

    *Helper method to set up context menu visual screen (drawable).*
- void __setUpVisualScreenFrame (void)

    *Helper method to set up framing for context menu visual screen (drawable).*
- void __drawVisualScreenFrame (void)

    *Helper method to draw visual screen frame.*
- void __setUpConsoleScreen (void)

    *Helper method to set up context menu console screen (drawable).*
- void __setUpConsoleScreenFrame (void)

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

- void __drawConsoleScreenFrame (void)

    *Helper method to draw console screen frame.*

- void __setConsoleState (ConsoleState)

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

- void __setConsoleString (void)

    *Helper method to set console string depending on console state.*

- void __drawConsoleText (void)

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

- void __handleKeyPressEvents (void)

    *Helper method to handle key press events.*

- void __handleMouseButtonEvents (void)

    *Helper method to handle mouse button events.*

- void __sendQuitGameMessage (void)

    *Helper method to format and send a quit game message.*

- void __sendRestartGameMessage (void)

    *Helper method to format and send a restart game message.*

## Private Attributes

- sf::Event ∗ event_ptr

    *A pointer to the event class.*

- sf::RenderWindow ∗ render_window_ptr

    *A pointer to the render window.*

- AssetsManager ∗ assets_manager_ptr

    *A pointer to the assets manager.*

- MessageHub ∗ message_hub_ptr

    *A pointer to the message hub.*

### 3.2.1 Detailed Description

A class which defines a context menu for the game.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 ContextMenu()

```
ContextMenu::ContextMenu (
            sf::Event * event_ptr,
            sf::RenderWindow * render_window_ptr,
            AssetsManager * assets_manager_ptr,
            MessageHub * message_hub_ptr )
```

Constructor for the ContextMenu class.

**Parameters**

| | |
|---|---|
| *event_ptr* | Pointer to the event class. |
| *render_window_ptr* | Pointer to the render window. |
| *assets_manager_ptr* | Pointer to the assets manager. |
| *message_hub_ptr* | Pointer to the message hub. |

```
782 {
783     //  1. set attributes
784
785     //  1.1. private
786     this->event_ptr = event_ptr;
787     this->render_window_ptr = render_window_ptr;
788
789     this->assets_manager_ptr = assets_manager_ptr;
790     this->message_hub_ptr = message_hub_ptr;
791
792     //  1.2. public
793     this->console_state = ConsoleState :: NONE_STATE;
794     this->__setConsoleState(ConsoleState :: READY);
795
796     this->game_menu_up = false;
797
798     this->frame = 0;
799
800     this->position_x = GAME_WIDTH;
801     this->position_y = 0;
802
803     //  2. set up and position drawable attributes
804     this->__setUpMenuFrame();
805     this->__setUpVisualScreen();
806     this->__setUpVisualScreenFrame();
807     this->__setUpConsoleScreen();
808     this->__setUpConsoleScreenFrame();
809
810     std::cout « "ContextMenu constructed at " « this « std::endl;
811
812     return;
813 }   /* ContextMenu() */
```

### 3.2.2.2 ∼ContextMenu()

```
ContextMenu::∼ContextMenu (
            void  )
```

Destructor for the ContextMenu class.
```
952 {
953     std::cout « "ContextMenu at " « this « " destroyed" « std::endl;
954
955     return;
956 }   /* ~ContextMenu() */
```

## 3.2.3 Member Function Documentation

### 3.2.3.1 __drawConsoleScreenFrame()

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

Helper method to draw console screen frame.
```
433 {
```

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

### 3.2.3.2 __drawConsoleText()

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

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

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

### 3.2.3.3 __drawVisualScreenFrame()

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

Helper method to draw visual screen frame.

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

### 3.2.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

### 3.2.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

### 3.2.3.6 __sendQuitGameMessage()

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

Helper method to format and send a quit game message.

```
712 {
713     Message quit_game_message;
714
715     quit_game_message.channel = GAME_CHANNEL;
716     quit_game_message.subject = "quit game";
717
718     this->message_hub_ptr->sendMessage(quit_game_message);
719
720     return;
721 } /* __sendQuitGameMessage() */
```

### 3.2.3.7 __sendRestartGameMessage()

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

Helper method to format and send a restart game message.

```
736 {
737     Message restart_game_message;
738
739     restart_game_message.channel = GAME_CHANNEL;
740     restart_game_message.subject = "restart game";
741
742     this->message_hub_ptr->sendMessage(restart_game_message);
743
744     return;
745 } /* __sendRestartGameMessage() */
```

### 3.2.3.8 __setConsoleState()

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

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

**Parameters**

| | |
|---|---|
| *console_state* | The state (ConsoleState) to set the console to. |

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

### 3.2.3.9  __setConsoleString()

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

Helper method to set console string depending on console state.

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

### 3.2.3.10 __setUpConsoleScreen()

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

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

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

### 3.2.3.11 __setUpConsoleScreenFrame()

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

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

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

```
306               )
307           );
308           this->console_screen_frame_left.setPoint(
309               1,
310               sf::Vector2f(
311                   this->position_x - 350 - 16,
312                   this->position_y + GAME_HEIGHT - 50 - 340 - 16
313               )
314           );
315           this->console_screen_frame_left.setPoint(
316               2,
317               sf::Vector2f(
318                   this->position_x - 350 - 16,
319                   this->position_y + GAME_HEIGHT - 50 + 16
320               )
321           );
322           this->console_screen_frame_left.setPoint(
323               3,
324               sf::Vector2f(
325                   this->position_x - 350,
326                   this->position_y + GAME_HEIGHT - 50
327               )
328           );
329
330           this->console_screen_frame_left.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.setFillColor(VISUAL_SCREEN_FRAME_GREY);
411
412      this->console_screen_frame_right.setOutlineThickness(2);
413      this->console_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
414
415      this->console_screen_frame_right.move(2, 0);
416
417      return;
418 }   /* __setUpConsoleScreenFrame() */
```

### 3.2.3.12 __setUpMenuFrame()

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

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

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

### 3.2.3.13 __setUpVisualScreen()

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

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

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

### 3.2.3.14 \_\_setUpVisualScreenFrame()

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

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

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

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

### 3.2.3.15  draw()

```
void ContextMenu::draw (
            void  )
```

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

```
922 {
923     //  1. menu frame
924     this->render_window_ptr->draw(this->menu_frame);
925
926     //  2. visual screen
927     this->render_window_ptr->draw(this->visual_screen);
928     this->__drawVisualScreenFrame();
929
930     //  3. console screen
931     this->render_window_ptr->draw(this->console_screen);
932     this->__drawConsoleScreenFrame();
933     this->__drawConsoleText();
934
935     this->frame++;
936     return;
937 }   /* draw() */
```

### 3.2.3.16  processEvent()

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

Method to processEvent ContextMenu. To be called once per event.

```
828 {
```

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

### 3.2.3.17 processMessage()

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

Method to processMessage ContextMenu. To be called once per message.

```
853 {
854      switch (this->console_state) {
855          case (ConsoleState :: TILE): {
856              //  process no tile selected
857              if (not this->message_hub_ptr->isEmpty(NO_TILE_SELECTED_CHANNEL)) {
858                  Message no_tile_selected_message = this->message_hub_ptr->receiveMessage(
859                      NO_TILE_SELECTED_CHANNEL
860                  );
861
862                  if (no_tile_selected_message.subject == "no tile selected") {
863                      this->__setConsoleState(ConsoleState :: READY);
864                      this->message_hub_ptr->popMessage(NO_TILE_SELECTED_CHANNEL);
865                  }
866              }
867
868              //  process tile state
869              if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
870                  Message tile_state_message = this->message_hub_ptr->receiveMessage(
871                      TILE_STATE_CHANNEL
872                  );
873
874                  if (tile_state_message.subject == "tile state") {
875                      this->console_string = tile_state_message.string_payload;
876                      this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
877                  }
878              }
879
880              //  process tile selected (subsequent left clicks causing program to hang)
881              if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
882                  this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
883              }
884
885              break;
886          }
887
888          default: {
889              //  process tile selected
890              if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
891                  Message tile_selected_message = this->message_hub_ptr->receiveMessage(
892                      TILE_SELECTED_CHANNEL
893                  );
894
895                  if (tile_selected_message.subject == "tile selected") {
896                      this->__setConsoleState(ConsoleState :: TILE);
897                      this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
898                  }
899              }
900
901              break;
902          }
903      }
904
905      return;
906 }    /* processMessage() */
```

## 3.2.4 Member Data Documentation

#### 3.2.4.1 assets_manager_ptr

[AssetsManager](#)* ContextMenu::assets_manager_ptr  [private]

A pointer to the assets manager.

#### 3.2.4.2 console_screen

sf::RectangleShape ContextMenu::console_screen

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

#### 3.2.4.3 console_screen_frame_bottom

sf::ConvexShape ContextMenu::console_screen_frame_bottom

The bottom framing of the console screen.

#### 3.2.4.4 console_screen_frame_left

sf::ConvexShape ContextMenu::console_screen_frame_left

The left framing of the console screen.

#### 3.2.4.5 console_screen_frame_right

sf::ConvexShape ContextMenu::console_screen_frame_right

The right framing of the console screen.

#### 3.2.4.6 console_screen_frame_top

sf::ConvexShape ContextMenu::console_screen_frame_top

The top framing of the console screen.

**3.2.4.7 console_state**

[ConsoleState](#) ContextMenu::console_state

The current state of the console screen.

**3.2.4.8 console_string**

std::string ContextMenu::console_string

The string to be printed to the console screen.

**3.2.4.9 event_ptr**

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

A pointer to the event class.

**3.2.4.10 frame**

int ContextMenu::frame

The current frame of this object.

**3.2.4.11 game_menu_up**

bool ContextMenu::game_menu_up

Indicates whether or not the game menu is up.

**3.2.4.12 menu_frame**

sf::RectangleShape ContextMenu::menu_frame

The frame of the context menu.

**3.2.4.13 message_hub_ptr**

[MessageHub](MessageHub)* ContextMenu::message_hub_ptr [private]

A pointer to the message hub.

**3.2.4.14 position_x**

double ContextMenu::position_x

The position of the object.

**3.2.4.15 position_y**

double ContextMenu::position_y

The position of the object.

**3.2.4.16 render_window_ptr**

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

A pointer to the render window.

**3.2.4.17 visual_screen**

sf::RectangleShape ContextMenu::visual_screen

The context menu screen for visuals.

**3.2.4.18 visual_screen_frame_bottom**

sf::ConvexShape ContextMenu::visual_screen_frame_bottom

The bottom framing of the visual screen.

**3.2.4.19 visual_screen_frame_left**

`sf::ConvexShape ContextMenu::visual_screen_frame_left`

The left framing of the visual screen.

**3.2.4.20 visual_screen_frame_right**

`sf::ConvexShape ContextMenu::visual_screen_frame_right`

The right framing of the visual screen.

**3.2.4.21 visual_screen_frame_top**

`sf::ConvexShape ContextMenu::visual_screen_frame_top`

The top framing of the visual screen.

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

- header/ContextMenu.h
- source/ContextMenu.cpp

## 3.3 Game Class Reference

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

`#include <Game.h>`

Collaboration diagram for Game:
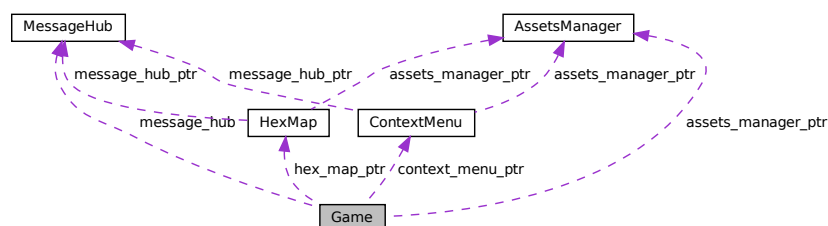
## Public Member Functions

- Game (sf::RenderWindow *, AssetsManager *)

  *Constructor for the Game class.*
- bool run (void)

  *Method to run game (defines game loop).*
- ∼Game (void)

  *Destructor for the Game class.*

## Public Attributes

- bool quit_game

  *Boolean indicating whether to quit (true) or create a new Game instance (false).*
- bool game_loop_broken

  *Boolean indicating whether or not the game loop is broken.*
- bool show_frame_clock_overlay

  *Boolean indicating whether or not to show frame and clock overlay.*
- unsigned long long int frame

  *The current frame of the game.*
- double time_since_start_s

  *The time elapsed [s] since the start of the game.*
- unsigned int year

  *Current game year.*
- unsigned int month

  *Current game month.*
- unsigned int population

  *Current population.*
- unsigned int credits

  *Current balance of credits.*
- unsigned int demand_MWh

  *Current energy demand [MWh].*
- unsigned int cumulative_emissions_tonnes

  *Cumulative emissions [tonnes] (1 tonne = 1000 kg).*
- sf::Clock clock

  *The game clock.*
- sf::Event event

  *The game events class.*
- MessageHub message_hub

  *The message hub (for inter-object message traffic).*
- HexMap * hex_map_ptr

  *Pointer to the hex map (defines game world).*
- ContextMenu * context_menu_ptr

  *Pointer to the context menu.*

## Private Member Functions

- void __toggleFrameClockOverlay (void)

    *Helper method to toggle frame clock overlay.*
- void __handleKeyPressEvents (void)

    *Helper method to handle key press events.*
- void __handleMouseButtonEvents (void)

    *Helper method to handle mouse button events.*
- void __processEvent (void)

    *Helper method to process Game. To be called once per event.*
- void __processMessage (void)

    *Helper method to process Game. To be called once per message.*
- void __drawFrameClockOverlay (void)

    *Helper method to draw frame clock overlay.*
- void __drawHUD (void)

    *Helper method to heads-up display (HUD).*
- void __draw (void)

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

## Private Attributes

- sf::RenderWindow ∗ render_window_ptr

    *A pointer to the render window.*
- AssetsManager ∗ assets_manager_ptr

    *A pointer to the assets manager.*

### 3.3.1 Detailed Description

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

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 Game()

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

Constructor for the Game class.

```
341 {
342     //  1. set attributes
343
344     //  1.1. private
345     this->render_window_ptr = render_window_ptr;
346
347     this->assets_manager_ptr = assets_manager_ptr;
348
349     //  1.2. public
350     this->quit_game = false;
```

```
351     this->game_loop_broken = false;
352     this->show_frame_clock_overlay = false;
353
354     this->frame = 0;
355     this->time_since_start_s = 0;
356
357     double seconds_since_epoch = time(NULL);
358     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
359
360     this->year = 1970 + (int)years_since_epoch;
361     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
362
363     this->population = 0;
364     this->credits = 0;
365     this->demand_MWh = 0;
366     this->cumulative_emissions_tonnes = 0;
367
368     this->hex_map_ptr = new HexMap(
369         6,
370         &(this->event),
371         this->render_window_ptr,
372         this->assets_manager_ptr,
373         &(this->message_hub)
374     );
375
376     this->context_menu_ptr = new ContextMenu(
377         &(this->event),
378         this->render_window_ptr,
379         this->assets_manager_ptr,
380         &(this->message_hub)
381     );
382
383     //  2. add message channel(s)
384     this->message_hub.addChannel(GAME_CHANNEL);
385
386     std::cout << "Game constructed at " << this << std::endl;
387
388     return;
389 }   /* Game() */
```

### 3.3.2.2 ∼Game()

```
Game::∼Game (
            void  )
```

Destructor for the Game class.

```
466 {
467     //  1. clean up attributes
468     delete this->hex_map_ptr;
469     delete this->context_menu_ptr;
470
471     std::cout << "Game at " << this << " destroyed" << std::endl;
472
473     return;
474 }   /* ∼Game() */
```

## 3.3.3 Member Function Documentation

### 3.3.3.1 __draw()

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

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

```
308 {
```

```
309     this->__drawHUD();
310
311     if (this->show_frame_clock_overlay) {
312         this->__drawFrameClockOverlay();
313     }
314
315     return;
316 }   /* draw() */
```

### 3.3.3.2 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.
```
200 {
201     std::string frame_clock_string = "FRAME: ";
202     frame_clock_string += std::to_string(this->frame);
203     frame_clock_string += "\nTIME SINCE START [s]: ";
204     frame_clock_string += std::to_string(this->time_since_start_s);
205
206     sf::Text frame_clock_text(
207         frame_clock_string,
208         *(this->assets_manager_ptr->getFont("DroidSansMono")),
209         16
210     );
211
212     sf::RectangleShape frame_clock_backing(
213         sf::Vector2f(
214             1.02 * frame_clock_text.getLocalBounds().width,
215             1.20 * frame_clock_text.getLocalBounds().height
216         )
217     );
218     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
219
220     this->render_window_ptr->draw(frame_clock_backing);
221     this->render_window_ptr->draw(frame_clock_text);
222
223     return;
224 }   /* __drawFrameClockOverlay() */
```

### 3.3.3.3 __drawHUD()

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

Helper method to heads-up display (HUD).
```
239 {
240     //  1. first line
241     std::string HUD_string = "YEAR: ";
242     HUD_string += std::to_string(this->year);
243
244     HUD_string += "    MONTH: ";
245     HUD_string += std::to_string(this->month);
246
247     HUD_string += "    POPULATION: ";
248     HUD_string += std::to_string(this->population);
249
250     HUD_string += "    CREDITS: ";
251     HUD_string += std::to_string(this->credits);
252     HUD_string += " K";
253
254     HUD_string += "    CURRENT DEMAND: ";
255     HUD_string += std::to_string(this->demand_MWh);
256     HUD_string += " MWh";
257
258     sf::Text HUD_text(
259         HUD_string,
260         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
```

```
261         16
262     );
263
264     HUD_text.setPosition(
265         (800 - HUD_text.getLocalBounds().width) / 2,
266         8
267     );
268
269     HUD_text.setFillColor(MONOCHROME_TEXT_GREEN);
270
271     this->render_window_ptr->draw(HUD_text);
272
273
274     //  2. second line
275     HUD_string = "CUMULATIVE EMISSIONS: ";
276     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
277     HUD_string += " tonnes (CO2e)";
278
279     HUD_string += "    LIFETIME LIMIT: ";
280     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
281     HUD_string += " tonnes (CO2e)";
282
283     HUD_text.setString(HUD_string);
284
285     HUD_text.setPosition(
286         (800 - HUD_text.getLocalBounds().width) / 2,
287         35
288     );
289
290     this->render_window_ptr->draw(HUD_text);
291
292     return;
293 }   /* __drawHUD() */
```

### 3.3.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
59 {
60     switch (this->event.key.code) {
61         case (sf::Keyboard::Tilde): {
62             this->__toggleFrameClockOverlay();
63
64             break;
65         }
66
67
68         case (sf::Keyboard::Tab): {
69             this->hex_map_ptr->toggleResourceOverlay();
70
71             break;
72         }
73
74
75         default: {
76             // do nothing!
77
78             break;
79         }
80     }
81
82     return;
83 }   /* __handleKeyPressEvents() */
```

### 3.3.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

### 3.3.3.6 __processEvent()

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

Helper method to process Game. To be called once per event.

```
138 {
139     if (this->event.type == sf::Event::Closed) {
140         this->quit_game = true;
141         this->game_loop_broken = true;
142     }
143
144     if (this->event.type == sf::Event::KeyPressed) {
145         this->__handleKeyPressEvents();
146     }
147
148     if (this->event.type == sf::Event::MouseButtonPressed) {
149         this->__handleMouseButtonEvents();
150     }
151
152     return;
153 }   /* __processEvent() */
```

### 3.3.3.7 __processMessage()

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

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

```
168 {
169     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
170         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
171
172         if (game_channel_message.subject == "quit game") {
173             this->quit_game = true;
174             this->game_loop_broken = true;
175             this->message_hub.popMessage(GAME_CHANNEL);
176         }
177
178         if (game_channel_message.subject == "restart game") {
179             this->game_loop_broken = true;
180             this->message_hub.popMessage(GAME_CHANNEL);
181         }
182     }
183
184     return;
185 }   /* __processMessage() */
```

### 3.3.3.8 __toggleFrameClockOverlay()

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

Helper method to toggle frame clock overlay.

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

### 3.3.3.9 run()

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

Method to run game (defines game loop).

**Returns**

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

```
407 {
408     //  1. play brand animation
409     //...
410
411     //  2. show splash screen
412     //...
413
414     //  3. start game loop
415     while (not this->game_loop_broken) {
416         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
417
418         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
419             //  6.1. process events
420             while (this->render_window_ptr->pollEvent(this->event)) {
421                 this->hex_map_ptr->processEvent();
422                 this->context_menu_ptr->processEvent();
423                 this->__processEvent();
424             }
425
426
427             //  6.2. process messages
428             while (this->message_hub.hasTraffic()) {
429                 this->hex_map_ptr->processMessage();
430                 this->context_menu_ptr->processMessage();
431                 this->__processMessage();
432             }
433
434
435             //  6.3. draw frame
436             this->render_window_ptr->clear();
437
438             this->hex_map_ptr->draw();
439             this->context_menu_ptr->draw();
440             this->__draw();
441
442             this->render_window_ptr->display();
443
444
445             //  6.4. increment frame
446             this->frame++;
447         }
448     }
449
450     return this->quit_game;
451 }   /* run() */
```

### 3.3.4 Member Data Documentation

#### 3.3.4.1 assets_manager_ptr

[AssetsManager](#)* Game::assets_manager_ptr  `[private]`

A pointer to the assets manager.

#### 3.3.4.2 clock

sf::Clock Game::clock

The game clock.

#### 3.3.4.3 context_menu_ptr

[ContextMenu](#)* Game::context_menu_ptr

Pointer to the context menu.

#### 3.3.4.4 credits

unsigned int Game::credits

Current balance of credits.

#### 3.3.4.5 cumulative_emissions_tonnes

unsigned int Game::cumulative_emissions_tonnes

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

### 3.3.4.6 demand_MWh

`unsigned int Game::demand_MWh`

Current energy demand [MWh].

### 3.3.4.7 event

`sf::Event Game::event`

The game events class.

### 3.3.4.8 frame

`unsigned long long int Game::frame`

The current frame of the game.

### 3.3.4.9 game_loop_broken

`bool Game::game_loop_broken`

Boolean indicating whether or not the game loop is broken.

### 3.3.4.10 hex_map_ptr

`HexMap* Game::hex_map_ptr`

Pointer to the hex map (defines game world).

### 3.3.4.11 message_hub

`MessageHub Game::message_hub`

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

**3.3.4.12 month**

`unsigned int Game::month`

Current game month.

**3.3.4.13 population**

`unsigned int Game::population`

Current population.

**3.3.4.14 quit_game**

`bool Game::quit_game`

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

**3.3.4.15 render_window_ptr**

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

A pointer to the render window.

**3.3.4.16 show_frame_clock_overlay**

`bool Game::show_frame_clock_overlay`

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

**3.3.4.17 time_since_start_s**

`double Game::time_since_start_s`

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

**3.3.4.18 year**

`unsigned int Game::year`

Current game year.

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

- header/Game.h
- source/Game.cpp

## 3.4 HexMap Class Reference

A class which defines a hex map of hex tiles.

`#include <HexMap.h>`

Collaboration diagram for HexMap:
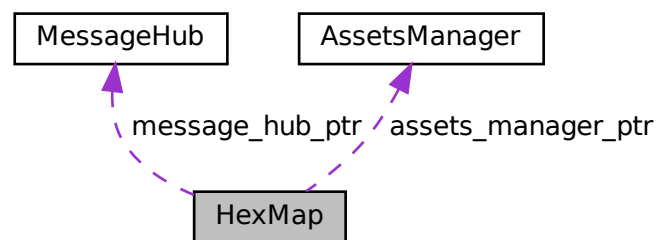


**Public Member Functions**

- HexMap (int, sf::Event ∗, sf::RenderWindow ∗, AssetsManager ∗, MessageHub ∗)

  *Constructor (intended) for the HexMap class.*
- void assess (void)

  *Method to assess the resource of the selected tile.*
- void reroll (void)

  *Method to re-roll the hex map.*
- void toggleResourceOverlay (void)

  *Method to toggle the hex map resource overlay.*
- void processEvent (void)

  *Method to process HexMap. To be called once per event.*
- void processMessage (void)

  *Method to process HexMap. To be called once per message.*
- void draw (void)

  *Method to draw the hex map to the render window. To be called once per frame.*
- void clear (void)

  *Method to clear the hex map.*
- ∼HexMap (void)

  *Destructor for the HexMap class.*

## Public Attributes

- bool tile_selected

    *A boolean which indicates if a tile is currently selected.*
- int n_layers

    *The number of layers in the hex map.*
- int n_tiles

    *The number of tiles in the hex map.*
- int frame

    *The current frame of this object.*
- double position_x

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

    *The y position of the hex map's origin (i.e. central) tile.*
- sf::RectangleShape glass_screen

    *To give the effect of an old glass screen over the hex map.*
- std::vector< double > tile_position_x_vec

    *A vector of tile x positions.*
- std::vector< double > tile_position_y_vec

    *A vector of tile y position.*
- std::vector< HexTile ∗ > border_tiles_vec

    *A vector of pointers to the border tiles.*
- std::map< double, std::map< double, HexTile ∗ > > hex_map

    *A position-indexed, nested map of hex tiles.*
- std::vector< HexTile ∗ > hex_draw_order_vec

    *A vector of hex tiles, in drawing order.*

## Private Member Functions

- void __setUpGlassScreen (void)

    *Helper method to set up glass screen effect (drawable).*
- void __layTiles (void)

    *Helper method to lay the hex tiles down to generate the game world.*
- void __buildDrawOrderVector (void)

    *Helper method to build tile drawing order vector.*
- std::vector< double > __getNoise (int, int=128)

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

    *Helper method to procedurally generate tile types and set tiles accordingly.*
- std::vector< double > __getValidMapIndexPositions (double, double)

    *Helper method to translate given position into valid index position for a.*
- std::vector< HexTile ∗ > __getNeighboursVector (HexTile ∗)

    *Helper method to assemble a vector pointers to all neighbours of the given tile.*
- TileType __getMajorityTileType (HexTile ∗)

    *Function to return majority tile type of a tile and its neighbours. If no clear majority, simply returns the type of the given tile.*
- void __smoothTileTypes (void)

    *Helper method to smooth tile types using a majority rules approach.*
- bool __isLakeTouchingOcean (HexTile ∗)
- void __enforceOceanContinuity (void)

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

- void __procedurallyGenerateTileResources (void)

  *Helper method to procedurally generate tile resources and set tiles accordingly.*
- void __assembleHexMap (void)

  *Helper method to assemble the hex map.*
- HexTile ∗ __getSelectedTile (void)

  *Helper method to get pointer to selected tile.*
- void __handleKeyPressEvents (void)

  *Helper method to handle key press events.*
- void __handleMouseButtonEvents (void)

  *Helper method to handle mouse button events.*
- void __sendNoTileSelectedMessage (void)

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

## Private Attributes

- sf::Event ∗ event_ptr

  *A pointer to the event class.*
- sf::RenderWindow ∗ render_window_ptr

  *A pointer to the render window.*
- AssetsManager ∗ assets_manager_ptr

  *A pointer to the assets manager.*
- MessageHub ∗ message_hub_ptr

  *A pointer to the message hub.*

### 3.4.1 Detailed Description

A class which defines a hex map of hex tiles.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 HexMap()

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

Constructor (intended) for the HexMap class.

**Parameters**

| | |
|---|---|
| *n_layers* | The number of layers in the HexMap. |
| *event_ptr* | Pointer to the event class. |
| *render_window_ptr* | Pointer to the render window. |
| *assets_manager_ptr* | Pointer to the assets manager. |
| *message_hub_ptr* | Pointer to the message hub. |

```
1056 {
1057     //  1. set attributes
1058
1059     //  1.1. private
1060     this->event_ptr = event_ptr;
1061     this->render_window_ptr = render_window_ptr;
1062
1063     this->assets_manager_ptr = assets_manager_ptr;
1064     this->message_hub_ptr = message_hub_ptr;
1065
1066     //  1.2. public
1067     this->tile_selected = false;
1068
1069     this->frame = 0;
1070
1071     this->n_layers = n_layers;
1072     if (this->n_layers < 0) {
1073         this->n_layers = 0;
1074     }
1075
1076     this->position_x = 400;
1077     this->position_y = 400;
1078
1079     //  2. assemble n layer hex map
1080     this->__assembleHexMap();
1081
1082     //  3. set up and position drawable attributes
1083     this->__setUpGlassScreen();
1084
1085     //  4. add message channel(s)
1086     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1087     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1088     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1089
1090     std::cout << "HexMap constructed at " << this << std::endl;
1091
1092     return;
1093 }   /* HexMap(), intended */
```

### 3.4.2.2 ∼HexMap()

```
HexMap::∼HexMap (
            void  )
```

Destructor for the HexMap class.

```
1345 {
1346     this->clear();
1347
1348     std::cout << "HexMap at " << this << " destroyed" << std::endl;
1349
1350     return;
1351 }   /* ~HexMap() */
```

## 3.4.3  Member Function Documentation

### 3.4.3.1  __assembleHexMap()

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

Helper method to assemble the hex map.

```
841 {
842     //  1. seed RNG (using milliseconds since 1 Jan 1970)
843     unsigned long long int milliseconds_since_epoch =
844         std::chrono::duration_cast<std::chrono::milliseconds>(
```

```
845            std::chrono::system_clock::now().time_since_epoch()
846        ).count();
847    srand(milliseconds_since_epoch);
848
849    //  2. lay tiles
850    this->__layTiles();
851    this->__buildDrawOrderVector();
852
853    //  3. procedurally generate types
854    this->__procedurallyGenerateTileTypes();
855
856    //  4. procedurally generate resources
857    this->__procedurallyGenerateTileResources();
858
859    return;
860 }  /* __assembleHexMap() */
```

### 3.4.3.2  __buildDrawOrderVector()

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

Helper method to build tile drawing order vector.

```
239 {
240    //  1. build temp list of tiles
241    std::list<HexTile*> temp_list;
242
243    std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
244    std::map<double, HexTile*>::iterator hex_map_iter_y;
245    for (
246        hex_map_iter_x = this->hex_map.begin();
247        hex_map_iter_x != this->hex_map.end();
248        hex_map_iter_x++
249    ) {
250        for (
251            hex_map_iter_y = hex_map_iter_x->second.begin();
252            hex_map_iter_y != hex_map_iter_x->second.end();
253            hex_map_iter_y++
254        ) {
255            temp_list.push_back(hex_map_iter_y->second);
256        }
257    }
258
259    //  2. move elements from temp list to drawing order vector
260    double min_position_y = 0;
261    std::list<HexTile*>::iterator list_iter;
262
263    while (not temp_list.empty()) {
264        //  2.1. determine min y position
265        min_position_y = std::numeric_limits<double>::infinity();
266
267        for (
268            list_iter = temp_list.begin();
269            list_iter != temp_list.end();
270            list_iter++
271        ) {
272            if ((*list_iter)->position_y < min_position_y) {
273                min_position_y = (*list_iter)->position_y;
274            }
275        }
276
277        //  2.2 move min y list elements to drawing order vec
278        list_iter = temp_list.begin();
279        while (list_iter != temp_list.end()) {
280            if ((*list_iter)->position_y == min_position_y) {
281                this->hex_draw_order_vec.push_back((*list_iter));
282                list_iter = temp_list.erase(list_iter);
283            }
284
285            else {
286                list_iter++;
287            }
288        }
289    }
290
291    return;
292 }  /* __buildDrawOrderVector() */
```

**3.4.3.3 __enforceOceanContinuity()**

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

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

```
752 {
753     std::cout « "enforcing ocean continuity ..." « std::endl;
754
755     bool tile_changed = false;
756
757     //  1. scan tiles and enforce (where appropriate)
758     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
759     std::map<double, HexTile*>::iterator hex_map_iter_y;
760     HexTile* hex_ptr;
761     for (
762         hex_map_iter_x = this->hex_map.begin();
763         hex_map_iter_x != this->hex_map.end();
764         hex_map_iter_x++
765     ) {
766         for (
767             hex_map_iter_y = hex_map_iter_x->second.begin();
768             hex_map_iter_y != hex_map_iter_x->second.end();
769             hex_map_iter_y++
770         ) {
771             hex_ptr = hex_map_iter_y->second;
772
773             if (this->__isLakeTouchingOcean(hex_ptr)) {
774                 hex_ptr->setTileType(TileType :: OCEAN);
775                 tile_changed = true;
776             }
777         }
778     }
779
780     if (tile_changed) {
781         this->__enforceOceanContinuity();
782     }
783     else {
784         return;
785     }
786 }   /* __enforceOceanContinuity() */
```

**3.4.3.4 __getMajorityTileType()**

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

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

**Parameters**

| | |
|---|---|
| *hex_ptr* | Pointer to the given tile. |

**Returns**

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

```
608 {
609     //  1. init type count map
610     std::map<TileType, int> type_count_map;
611     type_count_map[hex_ptr->tile_type] = 1;
612
613     //  2. survey neighbours, count type instances
```

```
614        std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
615
616        for (size_t i = 0; i < neighbours_vec.size(); i++) {
617            if (type_count_map.count(neighbours_vec[i]->tile_type) <= 0) {
618                type_count_map[neighbours_vec[i]->tile_type] = 1;
619            }
620            else {
621                type_count_map[neighbours_vec[i]->tile_type] += 1;
622            }
623        }
624
625        //  3. find majority tile type
626        int max_count = -1 * std::numeric_limits<int>::infinity();
627        TileType majority_tile_type = hex_ptr->tile_type;
628
629        std::map<TileType, int>::iterator map_iter;
630        for (
631            map_iter = type_count_map.begin();
632            map_iter != type_count_map.end();
633            map_iter++
634        ){
635            if (map_iter->second > max_count) {
636                max_count = map_iter->second;
637                majority_tile_type = map_iter->first;
638            }
639        }
640
641        //  4. detect ties
642        for (
643            map_iter = type_count_map.begin();
644            map_iter != type_count_map.end();
645            map_iter++
646        ){
647            if (
648                map_iter->second == max_count and
649                map_iter->first != majority_tile_type
650            ) {
651                majority_tile_type = hex_ptr->tile_type;
652                break;
653            }
654        }
655
656        return majority_tile_type;
657 }   /* __getMajorityTileType() */
```

### 3.4.3.5 __getNeighboursVector()

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

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

**Parameters**

| | |
|---|---|
| *hex_ptr* | A pointer to the given tile. |

**Returns**

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

```
550 {
551        std::vector<HexTile*> neighbours_vec;
552
553        //  1. build potential neighbour positions
554        std::vector<double> potential_neighbour_x_vec(6, 0);
555        std::vector<double> potential_neighbour_y_vec(6, 0);
556
557        for (int i = 0; i < 6; i++) {
558            potential_neighbour_x_vec[i] = hex_ptr->position_x +
559                2 * hex_ptr->minor_radius * cos((60 * i) * (M_PI / 180));
560
561            potential_neighbour_y_vec[i] = hex_ptr->position_y +
```

```
562            2 * hex_ptr->minor_radius * sin((60 * i) * (M_PI / 180));
563    }
564
565    //  2. populate neighbours vector
566    std::vector<double> map_index_positions;
567    double potential_x = 0;
568    double potential_y = 0;
569
570    for (int i = 0; i < 6; i++) {
571        potential_x = potential_neighbour_x_vec[i];
572        potential_y = potential_neighbour_y_vec[i];
573
574        map_index_positions = this->__getValidMapIndexPositions(
575            potential_x,
576            potential_y
577        );
578
579        if (not (map_index_positions[0] == -1)) {
580            neighbours_vec.push_back(
581                this->hex_map[map_index_positions[0]][map_index_positions[1]]
582            );
583        }
584    }
585
586    return neighbours_vec;
587 }   /* __getNeighbourVector() */
```

### 3.4.3.6  __getNoise()

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

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

**Parameters**

| | |
|---|---|
| *n_elements* | The number of elements in the generated noise vector. |
| *n_components* | The number of components to use in the random cosine series. Defaults to 64. |

**Returns**

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

```
315 {
316    //  1. generate random amplitude, wave number, direction, and phase vectors
317    std::vector<double> random_amplitude_vec(n_components, 0);
318    std::vector<double> random_wave_number_vec(n_components, 0);
319    std::vector<double> random_frequency_vec(n_components, 0);
320    std::vector<double> random_direction_vec(n_components, 0);
321    std::vector<double> random_phase_vec(n_components, 0);
322
323    for (int i = 0; i < n_components; i++) {
324        random_amplitude_vec[i] = 10 * ((double)rand() / RAND_MAX);
325
326        random_wave_number_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
327
328        random_frequency_vec[i] = ((double)rand() / RAND_MAX);
329
330        random_direction_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
331
332        random_phase_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
333    }
334
335    //  2. generate noise vec
336    double amp = 0;
337    double wave_no = 0;
338    double freq = 0;
339    double dir = 0;
```

```
340      double phase = 0;
341
342      double x = 0;
343      double y = 0;
344      double t = time(NULL);
345
346      double max_noise = -1 * std::numeric_limits<double>::infinity();
347      double min_noise = std::numeric_limits<double>::infinity();
348
349      double noise = 0;
350      std::vector<double> noise_vec(n_elements, 0);
351
352      for (int i = 0; i < n_elements; i++) {
353          x = this->tile_position_x_vec[i] - this->position_x;
354          y = this->tile_position_y_vec[i] - this->position_y;
355
356          for (int j = 0; j < n_components; j++) {
357              amp = random_amplitude_vec[j];
358              wave_no = random_wave_number_vec[j];
359              freq = random_frequency_vec[j];
360              dir = random_direction_vec[j];
361              phase = random_phase_vec[j];
362
363              noise += (amp / (j + 1)) * cos(
364                  wave_no * (j + 1) * (x * sin(dir) + y * cos(dir)) +
365                  2 * M_PI * (j + 1) * freq * t +
366                  phase
367              );
368          }
369
370          noise_vec[i] = noise;
371
372          if (noise > max_noise) {
373              max_noise = noise;
374          }
375
376          else if (noise < min_noise) {
377              min_noise = noise;
378          }
379
380          noise = 0;
381      }
382
383      // 3. normalize noise vec
384      for (int i = 0; i < n_elements; i++) {
385          noise_vec[i] = (noise_vec[i] - min_noise) / (max_noise - min_noise);
386
387          if (noise_vec[i] < 0) {
388              noise_vec[i] = 0;
389          }
390          else if (noise_vec[i] > 1) {
391              noise_vec[i] = 1;
392          }
393      }
394
395      return noise_vec;
396 }   /* __getNoise() */
```

### 3.4.3.7 __getSelectedTile()

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

Helper method to get pointer to selected tile.

**Returns**

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

```
877 {
878      HexTile* selected_tile_ptr = NULL;
879
880      bool break_flag = false;
881      std::map<double, std::map<double, HexTile*>>::iterator hex_map_iter_x;
882      std::map<double, HexTile*>::iterator hex_map_iter_y;
883
```

```
884      for (
885          hex_map_iter_x = this->hex_map.begin();
886          hex_map_iter_x != this->hex_map.end();
887          hex_map_iter_x++
888      ) {
889          for (
890              hex_map_iter_y = hex_map_iter_x->second.begin();
891              hex_map_iter_y != hex_map_iter_x->second.end();
892              hex_map_iter_y++
893          ) {
894              if (hex_map_iter_y->second->is_selected) {
895                  selected_tile_ptr = hex_map_iter_y->second;
896                  break_flag = true;
897              }
898
899              if (break_flag) {
900                  break;
901              }
902          }
903
904          if (break_flag) {
905              break;
906          }
907      }
908
909      return selected_tile_ptr;
910 }   /* __getSelectedTile() */
```

### 3.4.3.8 __getValidMapIndexPositions()

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

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

**Parameters**

| | |
|---|---|
| *potential↩_x* | The potential x position of the tile. |
| *potential↩_y* | The potential y position of the tile. |

**Returns**

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

```
496 {
497     std::vector<double> map_index_positions = {-1, -1};
498
499     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
500     std::map<double, HexTile*>::iterator hex_map_iter_y;
501     HexTile* hex_ptr;
502
503     double distance = 0;
504
505     for (
506         hex_map_iter_x = this->hex_map.begin();
507         hex_map_iter_x != this->hex_map.end();
508         hex_map_iter_x++
509     ) {
510         for (
511             hex_map_iter_y = hex_map_iter_x->second.begin();
512             hex_map_iter_y != hex_map_iter_x->second.end();
513             hex_map_iter_y++
514         ) {
515             hex_ptr = hex_map_iter_y->second;
516
517             distance = sqrt(
```

```
518                    pow(hex_ptr->position_x - potential_x, 2) +
519                    pow(hex_ptr->position_y - potential_y, 2)
520                );
521
522                if (distance <= hex_ptr->minor_radius / 4) {
523                    map_index_positions = {hex_ptr->position_x, hex_ptr->position_y};
524                    return map_index_positions;
525                }
526            }
527        }
528
529        return map_index_positions;
530 }    /* __isInHexMap() */
```

### 3.4.3.9  __handleKeyPressEvents()

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

Helper method to handle key press events.

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

### 3.4.3.10  __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
955 {
956     switch (this->event_ptr->mouseButton.button) {
957         case (sf::Mouse::Left): {
958             HexTile* hex_ptr = this->__getSelectedTile();
959
960             if (hex_ptr != NULL) {
961                 this->tile_selected = true;
962             }
963
964             else if (this->tile_selected) {
965                 this->tile_selected = false;
966                 this->__sendNoTileSelectedMessage();
967             }
968
969             break;
970         }
971
972
973         case (sf::Mouse::Right): {
974             if (this->tile_selected) {
975                 this->tile_selected = false;
976                 this->__sendNoTileSelectedMessage();
977             }
978
979             break;
980         }
```

```
981
982
983          default: {
984              // do nothing!
985
986              break;
987          }
988      }
989
990      return;
991 }   /* __handleMouseButtonEvents() */
```

### 3.4.3.11 __isLakeTouchingOcean()

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

### 3.4.3.12 __layTiles()

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

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

```
54 {
55      this->n_tiles = 0;
56
57      //  1. add origin tile
58      HexTile* hex_ptr = new HexTile(
59          this->position_x,
60          this->position_y,
61          this->event_ptr,
62          this->render_window_ptr,
63          this->assets_manager_ptr,
64          this->message_hub_ptr
65      );
66
67      this->hex_map[hex_ptr->position_x][hex_ptr->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->event_ptr,
79              this->render_window_ptr,
80              this->assets_manager_ptr,
81              this->message_hub_ptr
82          );
83
```

```
84             this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
85             this->tile_position_x_vec.push_back(hex_ptr->position_x);
86             this->tile_position_y_vec.push_back(hex_ptr->position_y);
87             this->n_tiles++;
88
89             if (i == this->n_layers - 1) {
90                 this->border_tiles_vec.push_back(hex_ptr);
91             }
92
93             hex_ptr = new HexTile(
94                 this->position_x - 2 * (i + 1) * hex_ptr->minor_radius,
95                 this->position_y,
96                 this->event_ptr,
97                 this->render_window_ptr,
98                 this->assets_manager_ptr,
99                 this->message_hub_ptr
100            );
101
102            this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
103            this->tile_position_x_vec.push_back(hex_ptr->position_x);
104            this->tile_position_y_vec.push_back(hex_ptr->position_y);
105            this->n_tiles++;
106
107            if (i == this->n_layers - 1) {
108                this->border_tiles_vec.push_back(hex_ptr);
109            }
110        }
111
112
113        //  3. fill out subsequent rows (reflect across first row)
114        HexTile* first_row_left_tile = hex_ptr;
115
116        int offset_count = 1;
117
118        double x_offset = 0;
119        double y_offset = 0;
120
121        for (
122            int row_width = 2 * this->n_layers;
123            row_width > this->n_layers;
124            row_width--
125        ) {
126            //  3.1. upper row
127            x_offset = first_row_left_tile->position_x +
128                2 * offset_count * first_row_left_tile->minor_radius *
129                cos(60 * (M_PI / 180));
130
131            y_offset = first_row_left_tile->position_y -
132                2 * offset_count * first_row_left_tile->minor_radius *
133                sin(60 * (M_PI / 180));
134
135            hex_ptr = new HexTile(
136                x_offset,
137                y_offset,
138                this->event_ptr,
139                this->render_window_ptr,
140                this->assets_manager_ptr,
141                this->message_hub_ptr
142            );
143
144            this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
145            this->tile_position_x_vec.push_back(hex_ptr->position_x);
146            this->tile_position_y_vec.push_back(hex_ptr->position_y);
147            this->n_tiles++;
148
149            this->border_tiles_vec.push_back(hex_ptr);
150
151            for (int i = 1; i < row_width; i++) {
152                x_offset += 2 * first_row_left_tile->minor_radius;
153
154                hex_ptr = new HexTile(
155                    x_offset,
156                    y_offset,
157                    this->event_ptr,
158                    this->render_window_ptr,
159                    this->assets_manager_ptr,
160                    this->message_hub_ptr
161                );
162
163                this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
164                this->tile_position_x_vec.push_back(hex_ptr->position_x);
165                this->tile_position_y_vec.push_back(hex_ptr->position_y);
166                this->n_tiles++;
167
168                if (row_width == this->n_layers + 1 or i == row_width - 1) {
169                    this->border_tiles_vec.push_back(hex_ptr);
170                }
```

```
171          }
172
173          //  3.2. lower row
174          x_offset = first_row_left_tile->position_x +
175              2 * offset_count * first_row_left_tile->minor_radius *
176              cos(60 * (M_PI / 180));
177
178          y_offset = first_row_left_tile->position_y +
179              2 * offset_count * first_row_left_tile->minor_radius *
180              sin(60 * (M_PI / 180));
181
182          hex_ptr = new HexTile(
183              x_offset,
184              y_offset,
185              this->event_ptr,
186              this->render_window_ptr,
187              this->assets_manager_ptr,
188              this->message_hub_ptr
189          );
190
191          this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
192          this->tile_position_x_vec.push_back(hex_ptr->position_x);
193          this->tile_position_y_vec.push_back(hex_ptr->position_y);
194          this->n_tiles++;
195
196          this->border_tiles_vec.push_back(hex_ptr);
197
198          for (int i = 1; i < row_width; i++) {
199              x_offset += 2 * first_row_left_tile->minor_radius;
200
201              hex_ptr = new HexTile(
202                  x_offset,
203                  y_offset,
204                  this->event_ptr,
205                  this->render_window_ptr,
206                  this->assets_manager_ptr,
207                  this->message_hub_ptr
208              );
209
210              this->hex_map[hex_ptr->position_x][hex_ptr->position_y] = hex_ptr;
211              this->tile_position_x_vec.push_back(hex_ptr->position_x);
212              this->tile_position_y_vec.push_back(hex_ptr->position_y);
213              this->n_tiles++;
214
215              if (row_width == this->n_layers + 1 or i == row_width - 1) {
216                  this->border_tiles_vec.push_back(hex_ptr);
217              }
218          }
219
220          offset_count++;
221      }
222
223      return;
224 }   /* __layTiles() */
```

### 3.4.3.13  __procedurallyGenerateTileResources()

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

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

```
801 {
802      //  1. get random cosine series noise vec
803      std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
804
805      //  2. set tile resources based on random cosine series noise
806      int noise_idx = 0;
807
808      std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
809      std::map<double, HexTile*>::iterator hex_map_iter_y;
810      for (
811          hex_map_iter_x = this->hex_map.begin();
812          hex_map_iter_x != this->hex_map.end();
813          hex_map_iter_x++
814      ) {
815          for (
816              hex_map_iter_y = hex_map_iter_x->second.begin();
817              hex_map_iter_y != hex_map_iter_x->second.end();
```

```
818              hex_map_iter_y++
819          ) {
820              hex_map_iter_y->second->setTileResource(noise_vec[noise_idx]);
821              noise_idx++;
822          }
823      }
824
825      return;
826  }  /* __procedurallyGenerateTileResources() */
```

### 3.4.3.14  __procedurallyGenerateTileTypes()

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

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

```
411  {
412      //  1. get random cosine series noise vec
413      std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
414
415      //  2. set initial tile types based on either random cosine series noise or white
416      //     noise (decided by coin toss)
417      int noise_idx = 0;
418
419      std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
420      std::map<double, HexTile*>::iterator hex_map_iter_y;
421      for (
422          hex_map_iter_x = this->hex_map.begin();
423          hex_map_iter_x != this->hex_map.end();
424          hex_map_iter_x++
425      ) {
426          for (
427              hex_map_iter_y = hex_map_iter_x->second.begin();
428              hex_map_iter_y != hex_map_iter_x->second.end();
429              hex_map_iter_y++
430          ) {
431              if ((double)rand() / RAND_MAX > 0.5) {
432                  hex_map_iter_y->second->setTileType(noise_vec[noise_idx]);
433              }
434              else {
435                  hex_map_iter_y->second->setTileType((double)rand() / RAND_MAX);
436              }
437              noise_idx++;
438          }
439      }
440
441      //  3. smooth tile types (majority rules)
442      this->__smoothTileTypes();
443
444      //  4. set border tile type to ocean
445      for (size_t i = 0; i < this->border_tiles_vec.size(); i++) {
446          this->border_tiles_vec[i]->setTileType(TileType :: OCEAN);
447      }
448
449      //  5. enforce ocean continuity (i.e. all lake tiles touching ocean become ocean)
450      this->__enforceOceanContinuity();
451
452      //  6. decorate tiles
453      for (
454          hex_map_iter_x = this->hex_map.begin();
455          hex_map_iter_x != this->hex_map.end();
456          hex_map_iter_x++
457      ) {
458          for (
459              hex_map_iter_y = hex_map_iter_x->second.begin();
460              hex_map_iter_y != hex_map_iter_x->second.end();
461              hex_map_iter_y++
462          ) {
463              hex_map_iter_y->second->decorateTile();
464          }
465      }
466
467      return;
468  }  /* __procedurallyGenerateTileTypes() */
```

### 3.4.3.15 __sendNoTileSelectedMessage()

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

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

```
1006 {
1007     Message no_tile_selected_message;
1008
1009     no_tile_selected_message.channel = NO_TILE_SELECTED_CHANNEL;
1010     no_tile_selected_message.subject = "no tile selected";
1011
1012     this->message_hub_ptr->sendMessage(no_tile_selected_message);
1013
1014     return;
1015 }   /* __sendNoTileSelectedMessage() */
```

### 3.4.3.16 __setUpGlassScreen()

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

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

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

### 3.4.3.17 __smoothTileTypes()

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

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

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

**3.4.3.18 assess()**

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

Method to assess the resource of the selected tile.

```
1108 {
1109     HexTile* selected_tile_ptr = this->__getSelectedTile();
1110     if (selected_tile_ptr != NULL) {
1111         selected_tile_ptr->assess();
1112     }
1113
1114     return;
1115 }   /* assess() */
```

**3.4.3.19 clear()**

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

Method to clear the hex map.

```
1307 {
1308     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1309     std::map<double, HexTile*>::iterator hex_map_iter_y;
1310     for (
1311         hex_map_iter_x = this->hex_map.begin();
1312         hex_map_iter_x != this->hex_map.end();
1313         hex_map_iter_x++
1314     ) {
1315         for (
1316             hex_map_iter_y = hex_map_iter_x->second.begin();
1317             hex_map_iter_y != hex_map_iter_x->second.end();
1318             hex_map_iter_y++
1319         ) {
1320             delete hex_map_iter_y->second;
1321         }
1322     }
1323     this->hex_map.clear();
1324
1325     this->tile_position_x_vec.clear();
1326     this->tile_position_y_vec.clear();
1327     this->border_tiles_vec.clear();
1328
1329     return;
1330 }   /* clear() */
```

**3.4.3.20 draw()**

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

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

```
1264 {
1265     //  1. draw background
1266     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1267     glass_screen_colour.a = 255;
1268     this->glass_screen.setFillColor(glass_screen_colour);
1269
1270     this->render_window_ptr->draw(this->glass_screen);
1271
1272     //  2. draw tiles in drawing order
1273     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1274         this->hex_draw_order_vec[i]->draw();
1275     }
1276
1277     //  3. redraw selected tile
```

```
1278      HexTile* selected_tile_ptr = this->__getSelectedTile();
1279      if (selected_tile_ptr != NULL) {
1280          selected_tile_ptr->draw();
1281      }
1282
1283      //  4. draw glass screen
1284      glass_screen_colour = this->glass_screen.getFillColor();
1285      glass_screen_colour.a = 40;
1286      this->glass_screen.setFillColor(glass_screen_colour);
1287
1288      this->render_window_ptr->draw(this->glass_screen);
1289
1290      this->frame++;
1291      return;
1292  }   /* draw() */
```

### 3.4.3.21 processEvent()

```
void HexMap::processEvent (
            void  )
```

Method to process HexMap. To be called once per event.

```
1183 {
1184      //  1. process HexTile events
1185      std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1186      std::map<double, HexTile*>::iterator hex_map_iter_y;
1187      for (
1188          hex_map_iter_x = this->hex_map.begin();
1189          hex_map_iter_x != this->hex_map.end();
1190          hex_map_iter_x++
1191      ) {
1192          for (
1193              hex_map_iter_y = hex_map_iter_x->second.begin();
1194              hex_map_iter_y != hex_map_iter_x->second.end();
1195              hex_map_iter_y++
1196          ) {
1197              hex_map_iter_y->second->processEvent();
1198          }
1199      }
1200
1201      //  2. process HexMap events
1202      if (this->event_ptr->type == sf::Event::KeyPressed) {
1203          this->__handleKeyPressEvents();
1204      }
1205
1206      if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1207          this->__handleMouseButtonEvents();
1208      }
1209
1210      return;
1211  }   /* processEvent() */
```

### 3.4.3.22 processMessage()

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

Method to process HexMap. To be called once per message.

```
1226 {
1227      //  1. process HexTile messages
1228      std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1229      std::map<double, HexTile*>::iterator hex_map_iter_y;
1230      for (
1231          hex_map_iter_x = this->hex_map.begin();
1232          hex_map_iter_x != this->hex_map.end();
1233          hex_map_iter_x++
1234      ) {
1235          for (
1236              hex_map_iter_y = hex_map_iter_x->second.begin();
```

```
1237                    hex_map_iter_y != hex_map_iter_x->second.end();
1238                    hex_map_iter_y++
1239            ) {
1240                hex_map_iter_y->second->processMessage();
1241            }
1242    }
1243
1244    //  2. process HexMap messages
1245    //...
1246
1247    return;
1248 }   /* processMessage() */
```

### 3.4.3.23 reroll()

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

Method to re-roll the hex map.

```
1130 {
1131    this->clear();
1132    this->__assembleHexMap();
1133
1134    return;
1135 }   /* reroll() */
```

### 3.4.3.24 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
1150 {
1151    std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1152    std::map<double, HexTile*>::iterator hex_map_iter_y;
1153    for (
1154        hex_map_iter_x = this->hex_map.begin();
1155        hex_map_iter_x != this->hex_map.end();
1156        hex_map_iter_x++
1157    ) {
1158        for (
1159            hex_map_iter_y = hex_map_iter_x->second.begin();
1160            hex_map_iter_y != hex_map_iter_x->second.end();
1161            hex_map_iter_y++
1162        ) {
1163            hex_map_iter_y->second->toggleResourceOverlay();
1164        }
1165    }
1166
1167    return;
1168 }   /* toggleResourceOverlay() */
```

### 3.4.4 Member Data Documentation

#### 3.4.4.1 assets_manager_ptr

AssetsManager* HexMap::assets_manager_ptr  [private]

A pointer to the assets manager.

### 3.4.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

### 3.4.4.3 event_ptr

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

A pointer to the event class.

### 3.4.4.4 frame

`int HexMap::frame`

The current frame of this object.

### 3.4.4.5 glass_screen

`sf::RectangleShape HexMap::glass_screen`

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

### 3.4.4.6 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

### 3.4.4.7 hex_map

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

A position-indexed, nested map of hex tiles.

### 3.4.4.8 message_hub_ptr

[MessageHub](#)* HexMap::message_hub_ptr  [private]

A pointer to the message hub.

### 3.4.4.9 n_layers

int HexMap::n_layers

The number of layers in the hex map.

### 3.4.4.10 n_tiles

int HexMap::n_tiles

The number of tiles in the hex map.

### 3.4.4.11 position_x

double HexMap::position_x

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

### 3.4.4.12 position_y

double HexMap::position_y

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

### 3.4.4.13 render_window_ptr

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

A pointer to the render window.

**3.4.4.14  tile_position_x_vec**

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

A vector of tile x positions.

**3.4.4.15  tile_position_y_vec**

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

A vector of tile y position.

**3.4.4.16  tile_selected**

`bool HexMap::tile_selected`

A boolean which indicates if a tile is currently selected.

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

- header/HexMap.h
- source/HexMap.cpp

## 3.5  HexTile Class Reference

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

`#include <HexTile.h>`

Collaboration diagram for HexTile:
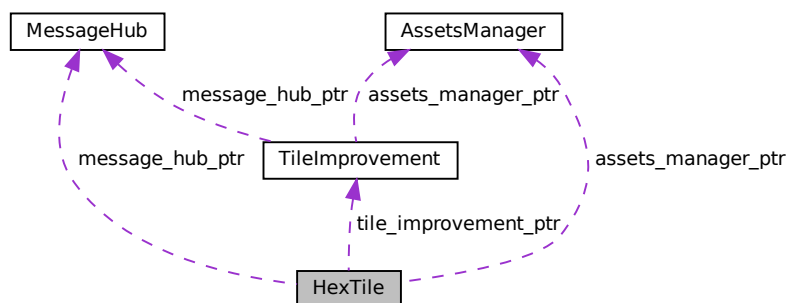
## Public Member Functions

- HexTile (double, double, sf::Event ∗, sf::RenderWindow ∗, AssetsManager ∗, MessageHub ∗)

     *Constructor for the HexTile class.*
- void setTileType (TileType)

     *Method to set the tile type (by enum value).*
- void setTileType (double)

     *Method to set the tile type (by numeric input).*
- void setTileResource (TileResource)

     *Method to set the tile resource (by enum value).*
- void setTileResource (double)

     *Method to set the tile resource (by numeric input).*
- void decorateTile (void)

     *Method to decorate tile.*
- void toggleResourceOverlay (void)

     *Method to toggle the tile resource overlay.*
- void assess (void)

     *Method to assess the tile's resource.*
- void processEvent (void)

     *Method to process HexTile. To be called once per event.*
- void processMessage (void)

     *Method to process HexTile. To be called once per message.*
- void draw (void)

     *Method to draw the hex tile to the render window. To be called once per frame.*
- ∼HexTile (void)

     *Destructor for the HexTile class.*

## Public Attributes

- TileType tile_type
- TileResource tile_resource
- bool show_node

     *A boolean which indicates whether or not to show the tile node.*
- bool show_resource

     *A boolean which indicates whether or not to show resource value.*
- bool resource_assessed

     *A boolean which indicates whether or not the resource has been assessed.*
- bool is_selected

     *A boolean which indicates whether or not the tile is selected.*
- bool settlement_built

     *A boolean which indicates if a settlement has been built yet or not.*
- bool has_improvement

     *A boolean which indicates if tile has improvement or not.*
- TileImprovement ∗ tile_improvement_ptr

     *A pointer to the improvement for this tile.*
- int frame

     *The current frame of this object.*
- double position_x

     *The x position of the tile.*
- double position_y

*The y position of the tile.*

- double major_radius

    *The radius of the smallest bounding circle.*

- double minor_radius

    *The radius of the largest inscribed circle.*

- sf::CircleShape node_sprite

    *A circle shape to mark the tile node.*

- sf::ConvexShape tile_sprite

    *A convex shape which represents the tile.*

- sf::ConvexShape select_outline_sprite

    *A convex shape which outlines the tile when selected.*

- sf::CircleShape resource_chip_sprite

    *A circle shape which represents a resource chip.*

- sf::Text resource_text

    *A text representation of the resource.*

- sf::Sprite tile_decoration_sprite

    *A tile decoration sprite.*

## Private Member Functions

- void __setUpNodeSprite (void)

    *Helper method to set up node sprite.*

- void __setUpTileSprite (void)

    *Helper method to set up tile sprite.*

- void __setUpSelectOutlineSprite (void)

    *Helper method to set up select outline sprite.*

- void __setUpResourceChipSprite (void)

    *Helper method to set up resource chip sprite.*

- void __setResourceText (void)

    *Helper method to set up resource text.*

- bool __isClicked (void)

    *Helper method to determine if tile was clicked on.*

- void __handleKeyPressEvents (void)

    *Helper method to handle key press events.*

- void __handleMouseButtonEvents (void)

    *Helper method to handle mouse button events.*

- void __sendTileSelectedMessage (void)

    *Helper method to format and send message on tile selection.*

- std::string __getTileCoordsSubstring (void)

    *Helper method to assemble and return tile coordinates substring.*

- std::string __getTileTypeSubstring (void)

    *Helper method to assemble and return tile type substring.*

- std::string __getTileResourceSubstring (void)

    *Helper method to assemble and return tile resource substring.*

- std::string __getTileImprovementSubstring (void)

    *Helper method to assemble and return the tile improvement substring.*

- std::string __getTileOptionsSubstring (void)

    *Helper method to assemble and return tile options substring.*

- void __sendTileStateMessage (void)

    *Helper method to format and send tile state message.*

**Private Attributes**

- sf::Event ∗ event_ptr

    *A pointer to the event class.*
- sf::RenderWindow ∗ render_window_ptr

    *A pointer to the render window.*
- AssetsManager ∗ assets_manager_ptr

    *A pointer to the assets manager.*
- MessageHub ∗ message_hub_ptr

    *A pointer to the message hub.*

### 3.5.1 Detailed Description

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

### 3.5.2 Constructor & Destructor Documentation

#### 3.5.2.1 HexTile()

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

Constructor for the HexTile class.

Ref: Wikipedia [2023]

**Parameters**

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

```
639 {
640     //  1. set attributes
641
642     //  1.1. private
643     this->event_ptr = event_ptr;
644     this->render_window_ptr = render_window_ptr;
645
646     this->assets_manager_ptr = assets_manager_ptr;
647     this->message_hub_ptr = message_hub_ptr;
648
```

```
649     //  1.2. public
650     this->show_node = false;
651     this->show_resource = false;
652     this->resource_assessed = false;
653     this->is_selected = false;
654
655     this->has_improvement = false;
656     this->settlement_built = false;
657     this->tile_improvement_ptr = NULL;
658
659     this->frame = 0;
660
661     this->position_x = position_x;
662     this->position_y = position_y;
663
664     this->major_radius = 32;
665     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
666
667     //  2. set up and position drawable attributes
668     this->__setUpNodeSprite();
669     this->__setUpTileSprite();
670     this->__setUpSelectOutlineSprite();
671     this->__setUpResourceChipSprite();
672     this->__setResourceText();
673
674     //  3. set tile type and resource (default to none type and average)
675     this->setTileType(TileType :: NONE_TYPE);
676     this->setTileResource(TileResource :: AVERAGE);
677
678     std::cout « "HexTile constructed at " « this « std::endl;
679
680     return;
681 }   /* HexTile() */
```

### 3.5.2.2 ∼HexTile()

```
HexTile::∼HexTile (
            void  )
```

Destructor for the HexTile class.

```
1125 {
1126     if (this->tile_improvement_ptr != NULL) {
1127         delete this->tile_improvement_ptr;
1128     }
1129
1130     std::cout « "HexTile at " « this « " destroyed" « std::endl;
1131
1132     return;
1133 }   /* ~HexTile() */
```

## 3.5.3 Member Function Documentation

### 3.5.3.1 __getTileCoordsSubstring()

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

Helper method to assemble and return tile coordinates substring.

**Returns**

Tile coordinates substring.

```
384 {
385     std::string coords_substring = "TILE COORDS:  (";
386     coords_substring += std::to_string(int(this->position_x - 400));
387     coords_substring += ", ";
388     coords_substring += std::to_string(int(this->position_y - 400));
389     coords_substring += ")\n";
390
391     return coords_substring;
392 } /* __getTileCoordsSubstring() */
```

### 3.5.3.2 __getTileImprovementSubstring()

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

Helper method to assemble and return the tile improvement substring.

**Returns**

Tile improvement substring.

```
511 {
512     std::string improvement_substring = "TILE IMPROVEMENT:  ";
513
514     if (this->has_improvement) {
515         //...
516     }
517
518     else {
519         improvement_substring += "NONE\n";
520     }
521
522     return improvement_substring;
523 } /* __getTileImprovementSubstring() */
```

### 3.5.3.3 __getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

**Returns**

Tile options substring.

```
540 {
541     //                    32 char x 17 line console "------------------------------\n";
542     std::string options_substring           = "    **** TILE OPTIONS ****    \n";
543     options_substring                       += "                             \n";
544
545     if ((not this->has_improvement) and (not this->settlement_built)) {
546         options_substring                   += "[B]:  BUILD SETTLEMENT       ";
547     }
548
549     return options_substring;
550 } /* __getTileOptionsString() */
```

### 3.5.3.4 __getTileResourceSubstring()

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

Helper method to assemble and return tile resource substring.

**Returns**

Tile resource substring.

```
473 {
474     std::string resource_substring = "TILE RESOURCE:     ";
475
476     if (this->resource_assessed) {
477         switch (this->tile_resource) {
478             //...
479
480
481             default: {
482                 resource_substring += "???\n";
483
484                 break;
485             }
486         }
487     }
488
489     else {
490         resource_substring += "[A]: ASSESS\n";
491     }
492
493     return resource_substring;
494 }   /* __getTileResourceSubstring() */
```

### 3.5.3.5 __getTileTypeSubstring()

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

Helper method to assemble and return tile type substring.

**Returns**

Tile type substring.

```
409 {
410     std::string type_substring = "TILE TYPE:         ";
411
412     switch (this->tile_type) {
413         case (TileType :: FOREST): {
414             type_substring += "FOREST\n";
415
416             break;
417         }
418
419
420         case (TileType :: LAKE): {
421             type_substring += "LAKE\n";
422
423             break;
424         }
425
426
427         case (TileType :: MOUNTAINS): {
428             type_substring += "MOUNTAINS\n";
429
430             break;
431         }
432
433
434         case (TileType :: OCEAN): {
```

```
435              type_substring += "OCEAN\n";
436
437          break;
438      }
439
440
441      case (TileType :: PLAINS): {
442          type_substring += "PLAINS\n";
443
444          break;
445      }
446
447
448      default: {
449          type_substring += "???\n";
450
451          break;
452      }
453  }
454
455  return type_substring;
456 }   /* __getTileTypeSubstring() */
```

### 3.5.3.6 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
277 {
278     switch (this->event_ptr->key.code) {
279         case (sf::Keyboard::Escape): {
280             this->is_selected = false;
281         }
282
283
284         default: {
285             // do nothing!
286
287             break;
288         }
289     }
290
291     return;
292 }   /* __handleKeyPressEvents() */
```

### 3.5.3.7 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
307 {
308     switch (this->event_ptr->mouseButton.button) {
309         case (sf::Mouse::Left): {
310             if (this->__isClicked()) {
311                 std::cout << "Tile (" << this->position_x << ", " <<
312                     this->position_y << ") was selected" << std::endl;
313
314                 this->is_selected = true;
315
316                 this->__sendTileSelectedMessage();
317                 this->__sendTileStateMessage();
318             }
319
320             else {
321                 this->is_selected = false;
322             }
323
```

```
324             break;
325         }
326
327
328         case (sf::Mouse::Right): {
329             this->is_selected = false;
330
331             break;
332         }
333
334
335         default: {
336             // do nothing!
337
338             break;
339         }
340     }
341
342     return;
343 }   /* __handleMouseButtonEvents() */
```

### 3.5.3.8 __isClicked()

```
bool HexTile::__isClicked (
            void  )  [private]
```

Helper method to determine if tile was clicked on.

**Returns**

Boolean indicating whether or not tile was clicked on.

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

### 3.5.3.9 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
358 {
359     Message tile_selected_message;
360
361     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
362     tile_selected_message.subject = "tile selected";
363
364     this->message_hub_ptr->sendMessage(tile_selected_message);
365
366     return;
367 }   /* __sendTileSelectedMessage() */
```

### 3.5.3.10 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
565 {
566     Message tile_state_message;
567
568     tile_state_message.channel = TILE_STATE_CHANNEL;
569     tile_state_message.subject = "tile state";
570
571
572     //                 32 char x 17 line console "------------------------------\n";
573     std::string string_payload                = "     **** TILE INFO ****      \n";
574     string_payload                           += "                              \n";
575
576     string_payload                           += this->__getTileCoordsSubstring();
577     string_payload                           += "                              \n";
578
579     string_payload                           += this->__getTileTypeSubstring();
580     string_payload                           += this->__getTileResourceSubstring();
581     string_payload                           += this->__getTileImprovementSubstring();
582     string_payload                           += "                              \n";
583
584     string_payload                           += this->__getTileOptionsSubstring();
585
586
587     tile_state_message.string_payload = string_payload;
588
589     this->message_hub_ptr->sendMessage(tile_state_message);
590
591     return;
592 }   /* __sendTileStateMessage() */
```

### 3.5.3.11 __setResourceText()

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

Helper method to set up resource text.

```
159 {
160     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
161
162     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
163
164     if (this->resource_assessed) {
165         switch (this->tile_resource) {
166             case (TileResource :: POOR): {
167                 this->resource_text.setString("-2");
168                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
169
170                 break;
171             }
172
173             case (TileResource :: BELOW_AVERAGE): {
174                 this->resource_text.setString("-1");
175                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
176
177                 break;
178             }
179
180             case (TileResource :: AVERAGE): {
181                 this->resource_text.setString("+0");
182
183                 break;
184             }
185
186             case (TileResource :: ABOVE_AVERAGE): {
187                 this->resource_text.setString("+1");
188                 this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
189
190                 break;
191             }
192
```

```
193                case (TileResource :: GOOD): {
194                    this->resource_text.setString("+2");
195                    this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
196
197                    break;
198                }
199
200                default: {
201                    this->resource_text.setString("?");
202
203                    break;
204                }
205            }
206        }
207
208        else {
209            this->resource_text.setString("?");
210        }
211
212        this->resource_text.setCharacterSize(20);
213
214        this->resource_text.setOrigin(
215            this->resource_text.getLocalBounds().width / 2,
216            this->resource_text.getLocalBounds().height / 2
217        );
218
219        this->resource_text.setPosition(
220            this->position_x,
221            this->position_y - 4
222        );
223
224        this->resource_text.setOutlineThickness(1);
225        this->resource_text.setOutlineColor(sf::Color(0, 0, 0, 255));
226
227        return;
228 }    /* __setResourceText() */
```

### 3.5.3.12 __setUpNodeSprite()

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

Helper method to set up node sprite.

```
34 {
35        this->node_sprite.setRadius(4);
36
37        this->node_sprite.setOrigin(
38            this->node_sprite.getLocalBounds().width / 2,
39            this->node_sprite.getLocalBounds().height / 2
40        );
41
42        this->node_sprite.setPosition(this->position_x, this->position_y);
43
44        this->node_sprite.setFillColor(sf::Color(255, 0, 0, 255));
45
46        return;
47 }    /* __setUpNodeSprite() */
```

### 3.5.3.13 __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, 250));
143
144      return;
145 }   /* __setUpResourceChip() */
```

### 3.5.3.14  __setUpSelectOutlineSprite()

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

Helper method to set up select outline sprite.

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

### 3.5.3.15  __setUpTileSprite()

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

Helper method to set up tile sprite.

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

### 3.5.3.16 assess()

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

Method to assess the tile's resource.

```
1000 {
1001     this->resource_assessed = true;
1002     this->__setResourceText();
1003     this->__sendTileStateMessage();
1004
1005     return;
1006 }   /* assess() */
```

### 3.5.3.17 decorateTile()

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

Method to decorate tile.

```
878 {
879     switch (this->tile_type) {
880         case (TileType :: FOREST): {
881             this->tile_decoration_sprite.setTexture(
882                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
883             );
884
885             break;
886         }
887
888         case (TileType :: LAKE): {
889             this->tile_decoration_sprite.setTexture(
890                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
891             );
892
893             break;
894         }
895
896         case (TileType :: MOUNTAINS): {
897             this->tile_decoration_sprite.setTexture(
898                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
899             );
900
901             break;
902         }
903
904         case (TileType :: OCEAN): {
905             this->tile_decoration_sprite.setTexture(
906                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
907             );
908
909             break;
910         }
911
912         case (TileType :: PLAINS): {
913             this->tile_decoration_sprite.setTexture(
914                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
915             );
916
917             break;
918         }
919
920         default: {
921             // do nothing!
922
923             break;
924         }
925     }
926
927
928     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
929         this->tile_decoration_sprite.setOrigin(
930             this->tile_decoration_sprite.getLocalBounds().width / 2,
931             this->tile_decoration_sprite.getLocalBounds().height / 2
932         );
```

```
933
934        this->tile_decoration_sprite.setPosition(
935            this->position_x,
936            this->position_y
937        );
938
939        if ((double)rand() / RAND_MAX > 0.5) {
940            this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
941        }
942    }
943
944    else {
945        this->tile_decoration_sprite.setOrigin(
946            this->tile_decoration_sprite.getLocalBounds().width / 2,
947            this->tile_decoration_sprite.getLocalBounds().height
948        );
949
950        this->tile_decoration_sprite.setPosition(
951            this->position_x,
952            this->position_y + 12
953        );
954
955        if ((double)rand() / RAND_MAX > 0.5) {
956            this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
957        }
958    }
959
960    return;
961 }    /* decorateTile(void) */
```

### 3.5.3.18  draw()

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

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

```
1078 {
1079     //  1. draw hex
1080     this->render_window_ptr->draw(this->tile_sprite);
1081
1082     //  2. draw node
1083     if (this->show_node) {
1084         this->render_window_ptr->draw(this->node_sprite);
1085     }
1086
1087     //  3. draw tile decoration
1088     this->render_window_ptr->draw(this->tile_decoration_sprite);
1089
1090     //  4. draw resource
1091     if (this->show_resource) {
1092         this->render_window_ptr->draw(this->resource_chip_sprite);
1093         this->render_window_ptr->draw(this->resource_text);
1094     }
1095
1096     //  5. draw selection outline
1097     if (this->is_selected) {
1098         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
1099
1100         outline_colour.a =
1101             255 * pow(cos((M_PI * this->frame) / (1.5 * FRAMES_PER_SECOND)), 2);
1102
1103         this->select_outline_sprite.setOutlineColor(outline_colour);
1104
1105         this->render_window_ptr->draw(this->select_outline_sprite);
1106     }
1107
1108     this->frame++;
1109     return;
1110 }    /* draw() */
```

**3.5.3.19 processEvent()**

```
void HexTile::processEvent (
             void  )
```

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

```
1021 {
1022     //  1. process TileImprovement events
1023     if (this->tile_improvement_ptr != NULL) {
1024         this->tile_improvement_ptr->processEvent();
1025     }
1026
1027     //  2. process HexTile events
1028     if (this->event_ptr->type == sf::Event::KeyPressed) {
1029         this->__handleKeyPressEvents();
1030     }
1031
1032     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1033         this->__handleMouseButtonEvents();
1034     }
1035
1036     return;
1037 }   /* processEvent() */
```

**3.5.3.20 processMessage()**

```
void HexTile::processMessage (
             void  )
```

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

```
1052 {
1053     //  1. process TileImprovement messages
1054     if (this->tile_improvement_ptr != NULL) {
1055         this->tile_improvement_ptr->processMessage();
1056     }
1057
1058     //  2. process HexTile messages
1059     //...
1060
1061     return;
1062 }   /* processMessage() */
```

**3.5.3.21 setTileResource()** [1/2]

```
void HexTile::setTileResource (
             double input_value )
```

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

**Parameters**

| *input_value* | A numerical input in the closed interval [0, 1]. |
| --- | --- |

```
827 {
828     //  1. check input
829     if (input_value < 0 or input_value > 1) {
830         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
831         error_str += "not in the closed interval [0, 1]";
832
833         #ifdef _WIN32
834             std::cout « error_str « std::endl;
```

```
835        #endif /* _WIN32 */
836
837        throw std::runtime_error(error_str);
838    }
839
840    // 2. convert input value to tile resource
841    TileResource tile_resource;
842
843    if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
844        tile_resource = TileResource :: POOR;
845    }
846    else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
847        tile_resource = TileResource :: BELOW_AVERAGE;
848    }
849    else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
850        tile_resource = TileResource :: AVERAGE;
851    }
852    else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
853        tile_resource = TileResource :: ABOVE_AVERAGE;
854    }
855    else {
856        tile_resource = TileResource :: GOOD;
857    }
858
859    // 3. call alternate method
860    this->setTileResource(tile_resource);
861
862    return;
863 }  /* setTileResource(double) */
```

### 3.5.3.22  setTileResource() [2/2]

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

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

**Parameters**

| | |
|---|---|
| *tile_resource* | The resource (TileResource) value to attribute to the tile. |

```
805 {
806    this->tile_resource = tile_resource;
807    this->__setResourceText();
808
809    return;
810 }  /* setTileResource(TileResource) */
```

### 3.5.3.23  setTileType() [1/2]

```
void HexTile::setTileType (
            double input_value )
```

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

**Parameters**

| | |
|---|---|
| *input_value* | A numerical input in the closed interval [0, 1]. |

```
755 {
756    // 1. check input
757    if (input_value < 0 or input_value > 1) {
```

```
758         std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
759         error_str += "not in the closed interval [0, 1]";
760
761         #ifdef _WIN32
762             std::cout « error_str « std::endl;
763         #endif  /* _WIN32 */
764
765         throw std::runtime_error(error_str);
766     }
767
768     //  2. convert input value to tile type
769     TileType tile_type;
770
771     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
772         tile_type = TileType :: LAKE;
773     }
774     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
775         tile_type = TileType :: PLAINS;
776     }
777     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
778         tile_type = TileType :: FOREST;
779     }
780     else {
781         tile_type = TileType :: MOUNTAINS;
782     }
783
784     //  3. call alternate method
785     this->setTileType(tile_type);
786
787     return;
788 }   /* setTileType(double) */
```

### 3.5.3.24  setTileType() [2/2]

```
void HexTile::setTileType (
            TileType tile_type )
```

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

**Parameters**

| | |
|---|---|
| *tile_type* | The type (TileType) to set the tile to. |

```
696 {
697     this->tile_type = tile_type;
698
699     switch (this->tile_type) {
700         case (TileType :: FOREST): {
701             this->tile_sprite.setFillColor(FOREST_GREEN);
702
703             break;
704         }
705
706         case (TileType :: LAKE): {
707             this->tile_sprite.setFillColor(LAKE_BLUE);
708
709             break;
710         }
711
712         case (TileType :: MOUNTAINS): {
713             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
714
715             break;
716         }
717
718         case (TileType :: OCEAN): {
719             this->tile_sprite.setFillColor(OCEAN_BLUE);
720
721             break;
722         }
723
724         case (TileType :: PLAINS): {
725             this->tile_sprite.setFillColor(PLAINS_YELLOW);
726
727             break;
```

```
728            }
729
730        default: {
731            // do nothing!
732
733            break;
734        }
735    }
736
737    return;
738 }  /* setTileType(TileType) */
```

### 3.5.3.25  toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
976 {
977    if (this->show_resource) {
978        this->show_resource = false;
979    }
980    else {
981        this->show_resource = true;
982    }
983
984    return;
985 }  /* toggleResourceOverlay() */
```

## 3.5.4  Member Data Documentation

### 3.5.4.1  assets_manager_ptr

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

A pointer to the assets manager.

### 3.5.4.2  event_ptr

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

A pointer to the event class.

### 3.5.4.3  frame

```
int HexTile::frame
```

The current frame of this object.

**3.5.4.4 has_improvement**

`bool HexTile::has_improvement`

A boolean which indicates if tile has improvement or not.

**3.5.4.5 is_selected**

`bool HexTile::is_selected`

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

**3.5.4.6 major_radius**

`double HexTile::major_radius`

The radius of the smallest bounding circle.

**3.5.4.7 message_hub_ptr**

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

A pointer to the message hub.

**3.5.4.8 minor_radius**

`double HexTile::minor_radius`

The radius of the largest inscribed circle.

**3.5.4.9 node_sprite**

`sf::CircleShape HexTile::node_sprite`

A circle shape to mark the tile node.

### 3.5.4.10 position_x

`double HexTile::position_x`

The x position of the tile.

### 3.5.4.11 position_y

`double HexTile::position_y`

The y position of the tile.

### 3.5.4.12 render_window_ptr

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

A pointer to the render window.

### 3.5.4.13 resource_assessed

`bool HexTile::resource_assessed`

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

### 3.5.4.14 resource_chip_sprite

`sf::CircleShape HexTile::resource_chip_sprite`

A circle shape which represents a resource chip.

### 3.5.4.15 resource_text

`sf::Text HexTile::resource_text`

A text representation of the resource.

**3.5.4.16 select_outline_sprite**

`sf::ConvexShape HexTile::select_outline_sprite`

A convex shape which outlines the tile when selected.

**3.5.4.17 settlement_built**

`bool HexTile::settlement_built`

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

**3.5.4.18 show_node**

`bool HexTile::show_node`

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

**3.5.4.19 show_resource**

`bool HexTile::show_resource`

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

**3.5.4.20 tile_decoration_sprite**

`sf::Sprite HexTile::tile_decoration_sprite`

A tile decoration sprite.

**3.5.4.21 tile_improvement_ptr**

`TileImprovement* HexTile::tile_improvement_ptr`

A pointer to the improvement for this tile.

**3.5.4.22 tile_resource**

`TileResource HexTile::tile_resource`

**3.5.4.23 tile_sprite**

`sf::ConvexShape HexTile::tile_sprite`

A convex shape which represents the tile.

**3.5.4.24 tile_type**

`TileType HexTile::tile_type`

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

- header/HexTile.h
- source/HexTile.cpp

# 3.6 Message Struct Reference

A structure which defines a standard message format.

`#include <MessageHub.h>`

## Public Attributes

- std::string channel = ""

    *A string identifying the appropriate channel for this message.*
- std::string subject = ""

    *A string describing the message subject.*
- std::vector< bool > bool_payload_vec = {}

    *A vector <bool> payload.*
- std::vector< int > int_payload_vec = {}

    *A vector <int> payload.*
- std::vector< double > double_payload_vec = {}

    *A vector <double> payload.*
- std::string string_payload = ""

    *A string payload.*

## 3.6.1 Detailed Description

A structure which defines a standard message format.

### 3.6.2 Member Data Documentation

#### 3.6.2.1 bool_payload_vec

```
std::vector<bool> Message::bool_payload_vec = {}
```

A vector <bool> payload.

#### 3.6.2.2 channel

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

A string identifying the appropriate channel for this message.

#### 3.6.2.3 double_payload_vec

```
std::vector<double> Message::double_payload_vec = {}
```

A vector <double> payload.

#### 3.6.2.4 int_payload_vec

```
std::vector<int> Message::int_payload_vec = {}
```

A vector <int> payload.

#### 3.6.2.5 string_payload

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

A string payload.

**3.6.2.6 subject**

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

A string describing the message subject.

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

- header/ESC_core/MessageHub.h

## 3.7 MessageHub Class Reference

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

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

### Public Member Functions

- MessageHub (void)

    *Constructor for the MessageHub class.*
- bool hasTraffic (void)

    *Method to determine if there remains any message traffic.*
- void addChannel (std::string)

    *Method to add channel to message map.*
- void removeChannel (std::string)

    *Method to remove channel from message map.*
- void sendMessage (Message)

    *Method to send a message to the message map.*
- bool isEmpty (std::string)

    *Method to check if channel is empty.*
- Message receiveMessage (std::string)

    *Method to receive the latest message in the given channel.*
- void popMessage (std::string)

    *Method to pop latest message off of the given channel.*
- void clearMessages (void)

    *Method to clear messages from the MessageHub.*
- void clear (void)

    *Method to clear the MessageHub.*
- ∼MessageHub (void)

    *Destructor for the MessageHub class.*

### Private Attributes

- std::map< std::string, std::list< Message > > message_map

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

### 3.7.1 Detailed Description

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

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 MessageHub()

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

Constructor for the [MessageHub](#) class.

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

#### 3.7.2.2 ∼MessageHub()

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

Destructor for the [MessageHub](#) class.

```
386 {
387     this->clear();
388
389     std::cout « "MessageHub at " « this « " destroyed" « std::endl;
390
391     return;
392 }   /* ∼MessageHub() */
```

### 3.7.3 Member Function Documentation

#### 3.7.3.1 addChannel()

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

Method to add channel to message map.

**Parameters**

| | |
|---|---|
| *channel* | The key for the message channel being added. |

```
97 {
98     //  1. check if channel is in map (if so, throw error)
99     if (this->message_map.count(channel) > 0) {
100        std::string error_str = "ERROR  MessageHub::addChannel()  channel ";
101        error_str += channel;
102        error_str += " is already in message map";
103
104        #ifdef _WIN32
105            std::cout « error_str « std::endl;
106        #endif  /* _WIN32 */
107
108        throw std::runtime_error(error_str);
109    }
110
111    //  2. add channel to map
112    this->message_map[channel] = {};
113
114    return;
115 }   /* addChannel() */
```

### 3.7.3.2  clear()

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

Method to clear the MessageHub.

```
366 {
367
368    this->clearMessages();
369    this->message_map.clear();
370
371    return;
372 }   /* clear() */
```

### 3.7.3.3  clearMessages()

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

Method to clear messages from the MessageHub.

```
340 {
341    std::map<std::string, std::list<Message»::iterator map_iter;
342    for (
343        map_iter = this->message_map.begin();
344        map_iter != this->message_map.end();
345        map_iter++
346    ) {
347        map_iter->second.clear();
348    }
349
350    return;
351 }   /* clearMessages() */
```

### 3.7.3.4 hasTraffic()

```
bool MessageHub::hasTraffic (
            void  )
```

Method to determine if there remains any message traffic.

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

### 3.7.3.5 isEmpty()

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

Method to check if channel is empty.

**Parameters**

| channel | The key for the message channel being checked. |
|---------|------------------------------------------------|

**Returns**

A boolean indicating whether the channel is empty or not.

```
207 {
208     //  1. check if channel is in map (if not, throw error)
209     if (this->message_map.count(channel) <= 0) {
210         std::string error_str = "ERROR  MessageHub::isEmpty()  channel ";
211         error_str += channel;
212         error_str += " is not in message map";
213
214         #ifdef _WIN32
215             std::cout « error_str « std::endl;
216         #endif  /* _WIN32 */
217
218         throw std::runtime_error(error_str);
219     }
220
221     if (this->message_map[channel].empty()) {
222         return true;
223     }
224     else {
225         return false;
226     }
227 }   /* isEmpty() */
```

### 3.7.3.6 popMessage()

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

Method to pop latest message off of the given channel.

**Parameters**

| channel | The key for the message channel being popped. |
|---------|-----------------------------------------------|

```
294 {
295     //  1. check if channel is in map (if not, throw error)
296     if (this->message_map.count(channel) <= 0) {
297         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
298         error_str += channel;
299         error_str += " is not in message map";
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. check if channel is empty (if so, throw error)
309     if (this->message_map[channel].empty()) {
310         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
311         error_str += channel;
312         error_str += " is empty";
313
314         #ifdef _WIN32
315             std::cout « error_str « std::endl;
316         #endif  /* _WIN32 */
317
318         throw std::runtime_error(error_str);
319     }
320
321     //  3. pop message
322     this->message_map[channel].pop_back();
323
324     return;
325 }   /* popMessage() */
```

### 3.7.3.7  receiveMessage()

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

Method to receive the latest message in the given channel.

**Parameters**

| channel | The key for the message channel being received from. |
|---------|------------------------------------------------------|

**Returns**

> The latest message in the given channel.

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

```
263         error_str += channel;
264         error_str += " is empty";
265
266         #ifdef _WIN32
267             std::cout << error_str << std::endl;
268         #endif  /* _WIN32 */
269
270         throw std::runtime_error(error_str);
271     }
272
273     //  3. receive message
274     Message message = this->message_map[channel].back();
275
276     return message;
277 }   /* receiveMessage() */
```

### 3.7.3.8   removeChannel()

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

Method to remove channel from message map.

**Parameters**

| *channel* | The key for the message channel being removed. |
|-----------|------------------------------------------------|

```
132 {
133     //  1. check if channel is in map (if not, throw error)
134     if (this->message_map.count(channel) <= 0) {
135         std::string error_str = "ERROR  MessageHub::removeChannel()  channel ";
136         error_str += channel;
137         error_str += " is not in message map";
138
139         #ifdef _WIN32
140             std::cout << error_str << std::endl;
141         #endif  /* _WIN32 */
142
143         throw std::runtime_error(error_str);
144     }
145
146     //  2. remove channel from map
147     this->message_map[channel].clear();
148     this->message_map.erase(channel);
149
150     return;
151 }   /* removeChannel() */
```

### 3.7.3.9   sendMessage()

```
void MessageHub::sendMessage (
            Message message )
```

Method to send a message to the message map.

**Parameters**

| *message* | The message to be sent. |
|-----------|-------------------------|

```
168 {
169     //  1. check if channel is in map (if not, throw error)
170     std::string channel = message.channel;
```

```
171
172     if (this->message_map.count(channel) <= 0) {
173         std::string error_str = "ERROR  MessageHub::sendMessage()  channel ";
174         error_str += channel;
175         error_str += " is not in message map";
176
177         #ifdef _WIN32
178             std::cout « error_str « std::endl;
179         #endif  /* _WIN32 */
180
181         throw std::runtime_error(error_str);
182     }
183
184     //  2. send message to message map
185     this->message_map[channel].push_back(message);
186
187     return;
188 }  /* sendMessage() */
```

### 3.7.4 Member Data Documentation

#### 3.7.4.1 message_map

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

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

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

- header/ESC_core/MessageHub.h
- source/ESC_core/MessageHub.cpp

## 3.8 TileImprovement Class Reference

A base class for the tile improvement hierarchy.

#include <TileImprovement.h>

Collaboration diagram for TileImprovement:
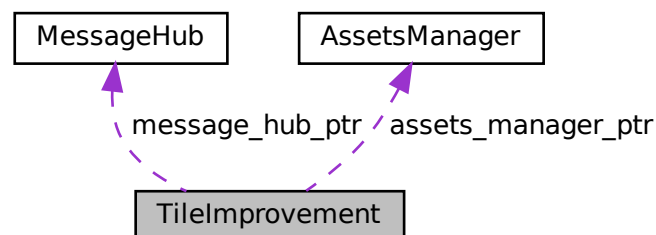
**Public Member Functions**

- TileImprovement (double, double, sf::Event ∗, sf::RenderWindow ∗, AssetsManager ∗, MessageHub ∗)

    *Constructor for the TileImprovement class.*
- virtual void processEvent (void)

    *Method to process TileImprovement. To be called once per event.*
- virtual void processMessage (void)

    *Method to process TileImprovement. To be called once per message.*
- virtual void draw (void)

    *Method to draw the hex tile to the render window. To be called once per frame.*
- virtual ∼TileImprovement (void)

    *Destructor for the TileImprovement class.*

**Public Attributes**

- int frame

    *The current frame of this object.*
- double position_x

    *The x position of the tile improvement.*
- double position_y

    *The y position of the tile improvement.*
- sf::Sprite tile_improvement_sprite_static

    *A static sprite, for decorating the tile.*
- std::vector< sf::Sprite > tile_improvement_sprite_animated

    *An animated sprite, for the ContextMenu visual screen.*

**Private Member Functions**

- virtual void __handleKeyPressEvents (void)

    *Helper method to handle key press events.*
- virtual void __handleMouseButtonEvents (void)

    *Helper method to handle mouse button events.*

**Private Attributes**

- sf::Event ∗ event_ptr

    *A pointer to the event class.*
- sf::RenderWindow ∗ render_window_ptr

    *A pointer to the render window.*
- AssetsManager ∗ assets_manager_ptr

    *A pointer to the assets manager.*
- MessageHub ∗ message_hub_ptr

    *A pointer to the message hub.*

### 3.8.1 Detailed Description

A base class for the tile improvement hierarchy.

## 3.8.2 Constructor & Destructor Documentation

### 3.8.2.1 TileImprovement()

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

Constructor for the TileImprovement class.

Ref: Wikipedia [2023]

**Parameters**

| | |
|---|---|
| *position_x* | The x position of the tile. |
| *position_y* | The y position of the tile. |
| *event_ptr* | Pointer to the event class. |
| *render_window_ptr* | Pointer to the render window. |
| *assets_manager_ptr* | Pointer to the assets manager. |
| *message_hub_ptr* | Pointer to the message hub. |

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

### 3.8.2.2 ∼TileImprovement()

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

Destructor for the TileImprovement class.
```
232 {
233     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
234
235     return;
236 }   /* ~TileImprovement() */
```

### 3.8.3 Member Function Documentation

#### 3.8.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

#### 3.8.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

#### 3.8.3.3 draw()

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

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

```
212 {
213      //...
214
215      this->frame++;
216      return;
217 }   /* draw() */
```

### 3.8.3.4 processEvent()

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

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

```
167 {
168     if (this->event_ptr->type == sf::Event::KeyPressed) {
169         this->__handleKeyPressEvents();
170     }
171
172     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
173         this->__handleMouseButtonEvents();
174     }
175
176     return;
177 }   /* processEvent() */
```

### 3.8.3.5 processMessage()

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

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

```
192 {
193     //...
194
195     return;
196 }   /* processMessage() */
```

## 3.8.4 Member Data Documentation

### 3.8.4.1 assets_manager_ptr

[AssetsManager]* TileImprovement::assets_manager_ptr [private]

A pointer to the assets manager.

### 3.8.4.2 event_ptr

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

A pointer to the event class.

**3.8.4.3 frame**

```
int TileImprovement::frame
```

The current frame of this object.

**3.8.4.4 message_hub_ptr**

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

A pointer to the message hub.

**3.8.4.5 position_x**

```
double TileImprovement::position_x
```

The x position of the tile improvement.

**3.8.4.6 position_y**

```
double TileImprovement::position_y
```

The y position of the tile improvement.

**3.8.4.7 render_window_ptr**

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

A pointer to the render window.

**3.8.4.8 tile_improvement_sprite_animated**

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

An animated sprite, for the ContextMenu visual screen.

**3.8.4.9 tile_improvement_sprite_static**

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

A static sprite, for decorating the tile.

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

- header/TileImprovement.h
- source/TileImprovement.cpp

# Chapter 4

# File Documentation

## 4.1 header/ContextMenu.h File Reference

Header file for the ContextMenu class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```
Include dependency graph for ContextMenu.h:
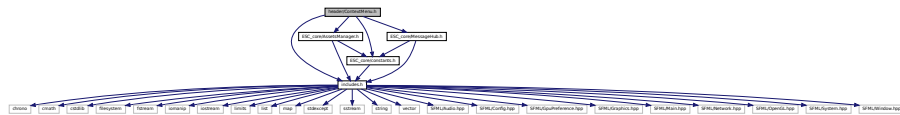


This graph shows which files directly or indirectly include this file:
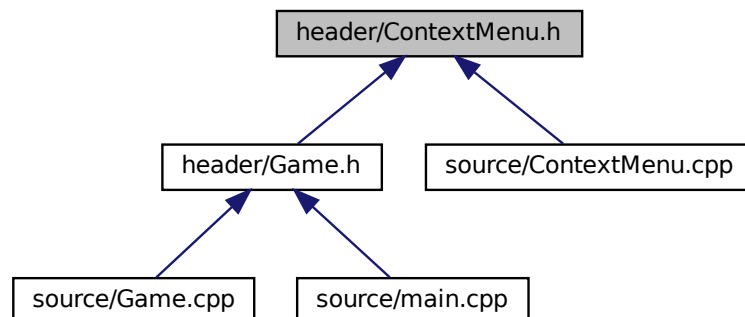
## Classes

- class ContextMenu

  *A class which defines a context menu for the game.*

## Enumerations

- enum ConsoleState {
  NONE_STATE , READY , MENU , TILE ,
  N_CONSOLE_STATES }

  *An enumeration of the different console screen states.*

### 4.1.1 Detailed Description

Header file for the ContextMenu class.

### 4.1.2 Enumeration Type Documentation

#### 4.1.2.1 ConsoleState

```
enum ConsoleState
```

An enumeration of the different console screen states.

**Enumerator**

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

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

## 4.2 header/ESC_core/AssetsManager.h File Reference

Header file for the AssetsManager class.

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

Include dependency graph for AssetsManager.h:
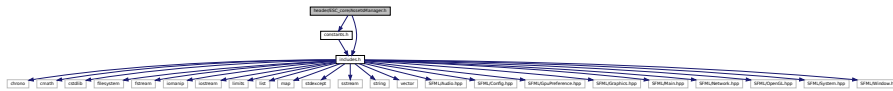


This graph shows which files directly or indirectly include this file:
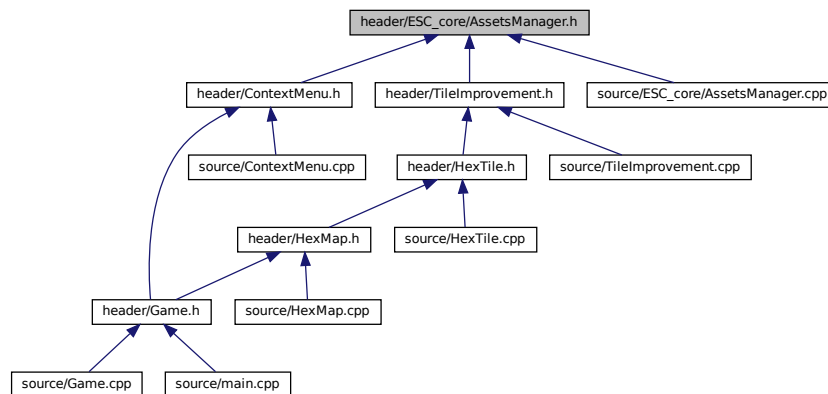


## Classes

- class AssetsManager

    *A class which manages visual and sound assets.*

### 4.2.1 Detailed Description

Header file for the AssetsManager class.

## 4.3 header/ESC_core/constants.h File Reference

Header file for various constants.

```
#include "includes.h"
```
Include dependency graph for constants.h:
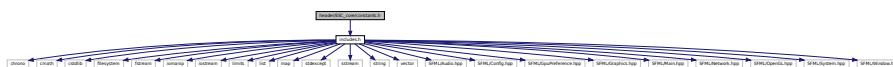
This graph shows which files directly or indirectly include this file:
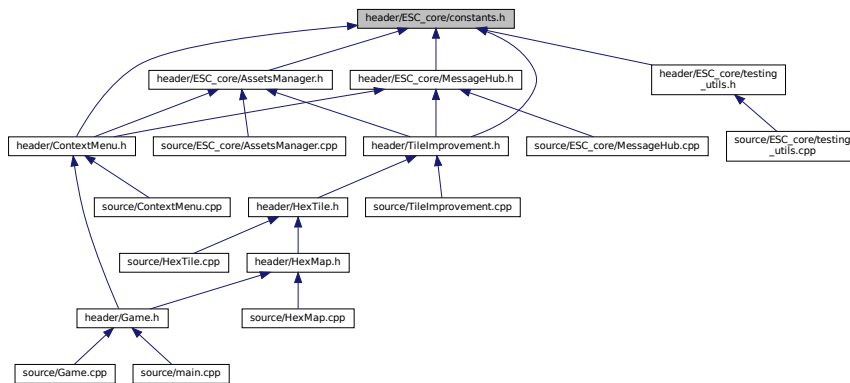


## Functions

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

    *The base colour of a forest tile.*

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

    *The base colour of a lake (water) tile.*

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

    *The base colour of a mountains tile.*

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

    *The base colour of an ocean (water) tile.*

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

    *The base colour of a plains tile.*

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

    *The base colour of the context menu frame.*

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

    *The base colour of old monochrome screens.*

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

    *The base colour of the framing of the visual screen.*

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

    *The base colour of old monochrome text (green).*

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

    *The base colour of old monochrome text (amber).*

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

    *The base colour of old monochrome text (red).*

## Variables

- const double FLOAT_TOLERANCE = 1e-6

    *Tolerance for floating point equality tests.*

- const unsigned long long int SECONDS_PER_YEAR = 31537970
- const unsigned long long int SECONDS_PER_MONTH = 2628164
- const int FRAMES_PER_SECOND = 60

    *Target frames per second.*

- const double SECONDS_PER_FRAME = 1.0 / 60

  *Target seconds per frame (just reciprocal of target frames per second).*
- const int GAME_WIDTH = 1200

  *Width of the game space.*
- const int GAME_HEIGHT = 800

  *Height of the game space.*
- const std::vector< double > TILE_TYPE_CUMULATIVE_PROBABILITIES

  *Cumulative probabilities for each tile type (to support procedural generation).*
- const std::vector< double > TILE_RESOURCE_CUMULATIVE_PROBABILITIES

  *Cumulative probabilities for each tile resource (to support procedural generation).*
- const std::string TILE_SELECTED_CHANNEL = "TILE SELECTED CHANNEL"

  *A message channel for tile selection messages.*
- const std::string NO_TILE_SELECTED_CHANNEL = "NO TILE SELECTED CHANNEL"

  *A message channel for no tile selected messages.*
- const std::string TILE_STATE_CHANNEL = "TILE STATE CHANNEL"

  *A message channel for tile state messages.*
- const unsigned int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500

  *The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.*
- const double CO2E_KG_PER_LITRE_DIESEL = 3.1596

  *The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.*
- const std::string GAME_CHANNEL = "GAME CHANNEL"

  *A message channel for game messages.*

## 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 CO2E_KG_PER_LITRE_DIESEL

```
const double CO2E_KG_PER_LITRE_DIESEL = 3.1596
```

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

#### 4.3.3.2 EMISSIONS_LIFETIME_LIMIT_TONNES

```
const unsigned int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500
```

The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.

#### 4.3.3.3 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

#### 4.3.3.4 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

#### 4.3.3.5 GAME_CHANNEL

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

A message channel for game messages.

#### 4.3.3.6 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

### 4.3.3.7 GAME_WIDTH

`const int GAME_WIDTH = 1200`

Width of the game space.

### 4.3.3.8 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

### 4.3.3.9 SECONDS_PER_FRAME

`const double SECONDS_PER_FRAME = 1.0 / 60`

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

### 4.3.3.10 SECONDS_PER_MONTH

`const unsigned long long int SECONDS_PER_MONTH = 2628164`

### 4.3.3.11 SECONDS_PER_YEAR

`const unsigned long long int SECONDS_PER_YEAR = 31537970`

### 4.3.3.12 TILE_RESOURCE_CUMULATIVE_PROBABILITIES

`const std::vector<double> TILE_RESOURCE_CUMULATIVE_PROBABILITIES`

**Initial value:**
```
= {
    0.10,
    0.30,
    0.70,
    0.90,
    1.00
}
```

Cumulative probabilities for each tile resource (to support procedural generation).

#### 4.3.3.13 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

#### 4.3.3.14 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

#### 4.3.3.15 TILE_TYPE_CUMULATIVE_PROBABILITIES

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

**Initial value:**
```
= {
    0.25,
    0.50,
    0.75,
    1.00
}
```

Cumulative probabilities for each tile type (to support procedural generation).

## 4.4 header/ESC_core/doxygen_cite.h File Reference

Header file which simply cites the doxygen tool.

### 4.4.1 Detailed Description

Header file which simply cites the doxygen tool.

Ref: van Heesch. [2023]

## 4.5 header/ESC_core/includes.h File Reference

Header file for various includes.

```
#include <chrono>
#include <cmath>
#include <cstdlib>
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <limits>
#include <list>
#include <map>
#include <stdexcept>
#include <sstream>
#include <string>
#include <vector>
#include <SFML/Audio.hpp>
#include <SFML/Config.hpp>
#include <SFML/GpuPreference.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Main.hpp>
#include <SFML/Network.hpp>
#include <SFML/OpenGL.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
```
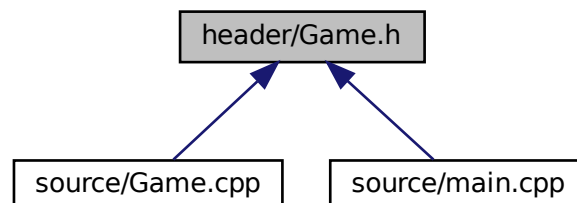Include dependency graph for includes.h:



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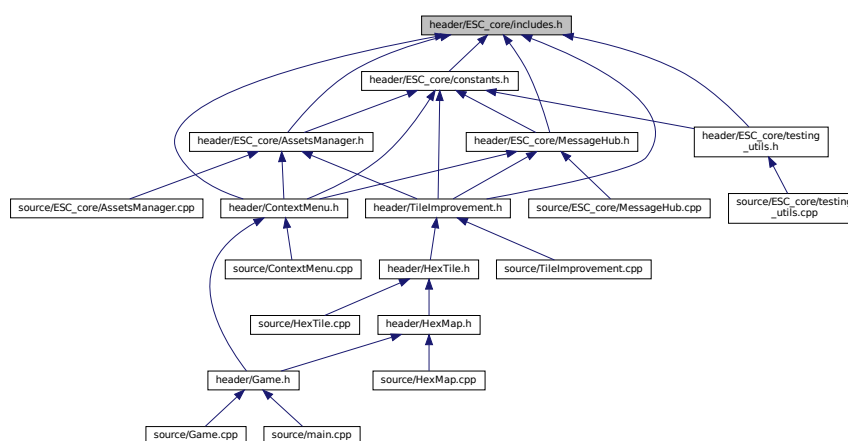


### 4.5.1 Detailed Description

Header file for various includes.

Ref: Gomila [2023]

## 4.6 header/ESC_core/MessageHub.h File Reference

Header file for the MessageHub class.

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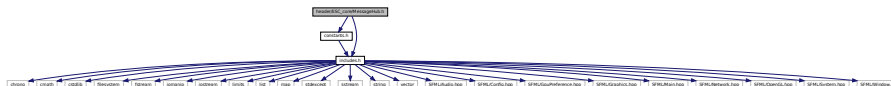


This graph shows which files directly or indirectly include this file:
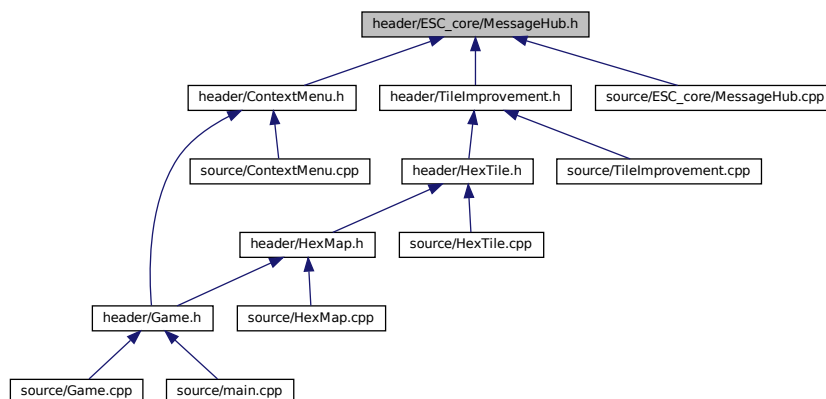


### Classes

- struct Message

    *A structure which defines a standard message format.*
- class MessageHub

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

### 4.6.1 Detailed Description

Header file for the MessageHub class.

## 4.7 header/ESC_core/testing_utils.h File Reference

Header file for various testing utilities.

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



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



## Functions

- void printGreen (std::string)

  *A function that sends green text to std::cout.*
- void printGold (std::string)

  *A function that sends gold text to std::cout.*
- void printRed (std::string)

  *A function that sends red text to std::cout.*
- void testFloatEquals (double, double, std::string, int)

  *Tests for the equality of two floating point numbers x and y (to within FLOAT_TOLERANCE).*
- void testGreaterThan (double, double, std::string, int)

  *Tests if $x > y$.*
- void testGreaterThanOrEqualTo (double, double, std::string, int)

  *Tests if $x >= y$.*
- void testLessThan (double, double, std::string, int)

  *Tests if $x < y$.*
- void testLessThanOrEqualTo (double, double, std::string, int)

  *Tests if $x <= y$.*
- void testTruth (bool, std::string, int)

  *Tests if the given statement is true.*
- void expectedErrorNotDetected (std::string, int)

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

### 4.7.1 Detailed Description

Header file for various testing utilities.

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

### 4.7.2 Function Documentation

#### 4.7.2.1 expectedErrorNotDetected()

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

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

**Parameters**

| | |
|---|---|
| *file* | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| *line* | The line of the file in which the test is applied (you should be able to just pass in "__LINE__"). |

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

#### 4.7.2.2 printGold()

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

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

**Parameters**

| | |
|---|---|
| *input_str* | The text of the string to be sent to std::cout. |

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

### 4.7.2.3 printGreen()

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

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

**Parameters**

| input_str | 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() */
```

### 4.7.2.4 printRed()

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

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

**Parameters**

| input_str | The text of the string to be sent to std::cout. |
|-----------|--------------------------------------------------|

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

### 4.7.2.5 testFloatEquals()

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

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

**Parameters**

| x    | The first of two numbers to test. |
|------|-----------------------------------|
| y    | The second of two numbers to test. |
| file | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| line | 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() */
```

### 4.7.2.6 testGreaterThan()

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

Tests if x > y.

**Parameters**

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

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

### 4.7.2.7 testGreaterThanOrEqualTo()

```
void testGreaterThanOrEqualTo (
            double x,
```

```
            double y,
            std::string file,
            int line )
```

Tests if x >= y.

**Parameters**

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

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

### 4.7.2.8 testLessThan()

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

Tests if x < y.

**Parameters**

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

```
291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
```

```
301        error_str += std::to_string(x);
302        error_str += " is not less than ";
303        error_str += std::to_string(y);
304        error_str += "\n";
305
306        #ifdef _WIN32
307            std::cout << error_str << std::endl;
308        #endif
309
310        throw std::runtime_error(error_str);
311        return;
312 }    /* testLessThan() */
```

### 4.7.2.9 testLessThanOrEqualTo()

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

Tests if x <= y.

**Parameters**

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

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

### 4.7.2.10 testTruth()

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

Tests if the given statement is true.

**Parameters**

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

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

## 4.8 header/Game.h File Reference

```
#include "HexMap.h"
#include "ContextMenu.h"
```
Include dependency graph for Game.h:
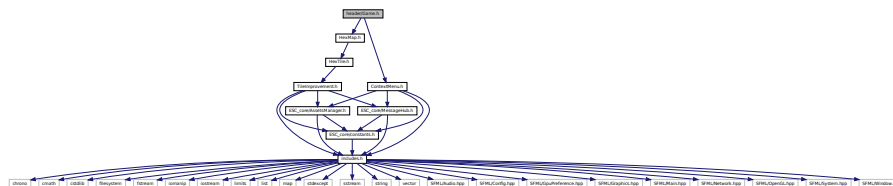


This graph shows which files directly or indirectly include this file:



## Classes

- class Game

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

## 4.9 header/HexMap.h File Reference

Header file for the HexMap class.

```
#include "HexTile.h"
```
Include dependency graph for HexMap.h:



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



### Classes

- class HexMap

  *A class which defines a hex map of hex tiles.*

### 4.9.1 Detailed Description

Header file for the HexMap class.

## 4.10 header/HexTile.h File Reference

Header file for the Game class.

```
#include "TileImprovement.h"
```
Include dependency graph for HexTile.h:
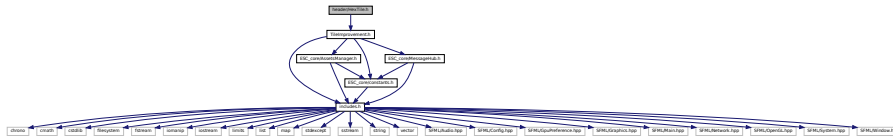


This graph shows which files directly or indirectly include this file:
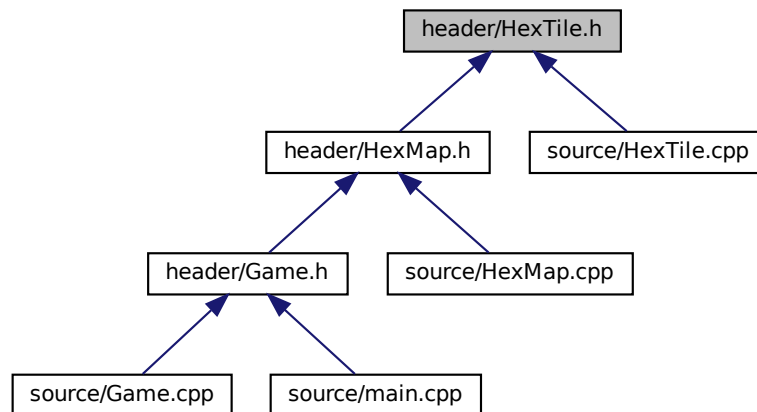


## Classes

- class HexTile

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

## Enumerations

- enum TileType {
  NONE_TYPE , FOREST , LAKE , MOUNTAINS ,
  OCEAN , PLAINS , N_TILE_TYPES }

    *An enumeration of the different tile types.*
- enum TileResource {
  POOR , BELOW_AVERAGE , AVERAGE , ABOVE_AVERAGE ,
  GOOD , N_TILE_RESOURCES }

    *An enumeration of the different tile resource values.*

### 4.10.1 Detailed Description

Header file for the Game class.

Header file for the HexTile class.

### 4.10.2 Enumeration Type Documentation

#### 4.10.2.1 TileResource

```
enum TileResource
```

An enumeration of the different tile resource values.

**Enumerator**

| | |
|---|---|
| POOR | A poor resource value. |
| BELOW_AVERAGE | A below average resource value. |
| AVERAGE | An average resource value. |
| ABOVE_AVERAGE | An above average resource value. |
| GOOD | A good resource value. |
| N_TILE_RESOURCES | A simple hack to get the number of elements in TileResource. |

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

#### 4.10.2.2 TileType

```
enum TileType
```

An enumeration of the different tile types.

**Enumerator**

| | |
|---|---|
| NONE_TYPE | A dummy tile (for initialization). |
| FOREST | A forest tile. |
| LAKE | A lake tile. |
| MOUNTAINS | A mountains tile. |
| OCEAN | An ocean tile. |
| PLAINS | A plains tile. |
| N_TILE_TYPES | A simple hack to get the number of elements in TileType. |

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

## 4.11 header/TileImprovement.h File Reference

Header file for the TileImprovement class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```
Include dependency graph for TileImprovement.h:
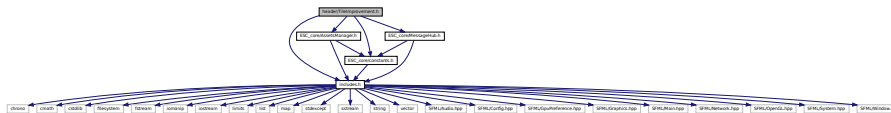


This graph shows which files directly or indirectly include this file:
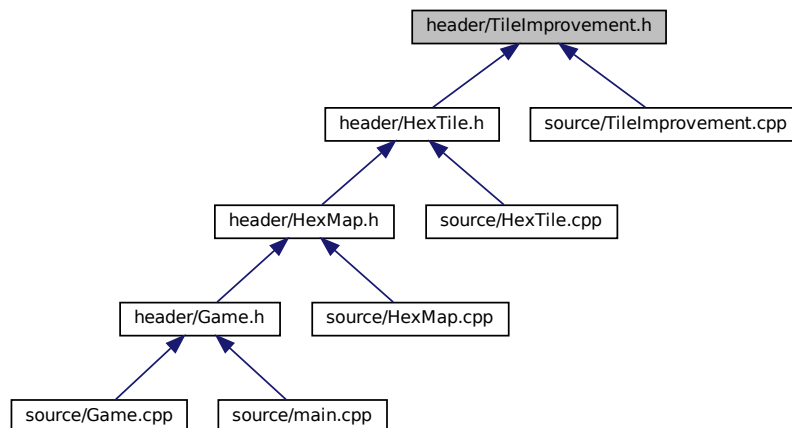


### Classes

- class TileImprovement

    *A base class for the tile improvement hierarchy.*

**Enumerations**

- enum TileImprovementType {
  SETTLEMENT , SOLAR_PV , WIND_TURBINE , TIDAL_TURBINE ,
  WAVE_ENERGY_CONVERTER , ENERGY_STORAGE_SYSTEM , N_TILE_IMPROVEMENT_TYPES }

  *An enumeration of the different tile improvement types.*

### 4.11.1 Detailed Description

Header file for the TileImprovement class.

### 4.11.2 Enumeration Type Documentation

#### 4.11.2.1 TileImprovementType

```
enum TileImprovementType
```

An enumeration of the different tile improvement types.

**Enumerator**

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

```
34                        {
35     SETTLEMENT,
36     SOLAR_PV,
37     WIND_TURBINE,
38     TIDAL_TURBINE,
39     WAVE_ENERGY_CONVERTER,
40     ENERGY_STORAGE_SYSTEM,
41     N_TILE_IMPROVEMENT_TYPES
42 }; /* TileImprovementType */
```

## 4.12 source/ContextMenu.cpp File Reference

Implementation file for the ContextMenu class.

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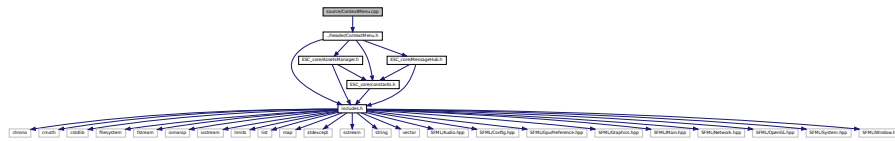


### 4.12.1 Detailed Description

Implementation file for the ContextMenu class.

A class which defines a context menu for the game.

## 4.13 source/ESC_core/AssetsManager.cpp File Reference

Implementation file for the AssetsManager class.

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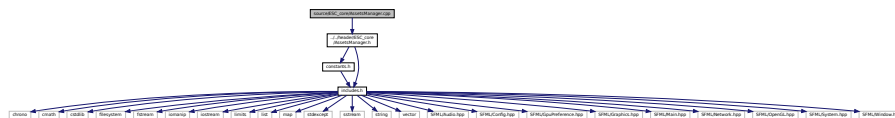


### 4.13.1 Detailed Description

Implementation file for the AssetsManager class.

A class which manages visual and sound assets.

## 4.14 source/ESC_core/MessageHub.cpp File Reference

Implementation file for the MessageHub class.

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

### 4.14.1 Detailed Description

Implementation file for the MessageHub class.

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

## 4.15 source/ESC_core/testing_utils.cpp File Reference

Implementation file for various testing utilities.

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



### Functions

- void printGreen (std::string input_str)

  *A function that sends green text to std::cout.*
- void printGold (std::string input_str)

  *A function that sends gold text to std::cout.*
- void printRed (std::string input_str)

  *A function that sends red text to std::cout.*
- void testFloatEquals (double x, double y, std::string file, int line)

  *Tests for the equality of two floating point numbers x and y (to within FLOAT_TOLERANCE).*
- void testGreaterThan (double x, double y, std::string file, int line)

  *Tests if x > y.*
- void testGreaterThanOrEqualTo (double x, double y, std::string file, int line)

  *Tests if x >= 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 <= 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**

| file | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
|------|------|
| line | 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**

| input_str | 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**

| | |
|---|---|
| *input_str* | 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**

| | |
|---|---|
| *input_str* | The text of the string to be sent to std::cout. |

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

### 4.15.2.5 testFloatEquals()

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

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

**Parameters**

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

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

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

### 4.15.2.6 testGreaterThan()

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

Tests if x > y.

**Parameters**

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

**Parameters**

| | |
|---|---|
| *x* | The first of two numbers to test. |
| *y* | The second of two numbers to test. |
| *file* | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| *line* | 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**

| | |
|---|---|
| *x* | The first of two numbers to test. |
| *y* | The second of two numbers to test. |
| *file* | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| *line* | 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 $<=$ y.

**Parameters**

| x | The first of two numbers to test. |
|---|---|
| y | The second of two numbers to test. |
| file | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| line | 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**

| statement | The statement whose truth is to be tested ("1 == 0", for example). |
|---|---|
| file | The file in which the test is applied (you should be able to just pass in "__FILE__"). |
| line | 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/Game.cpp File Reference

Implementation file for the Game class.

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



### 4.16.1 Detailed Description

Implementation file for the Game class.

A class which defines a tile of a hex map.

## 4.17 source/HexMap.cpp File Reference

Implementation file for the HexMap class.

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

### 4.17.1 Detailed Description

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

A class which defines a hex map of hex tiles.

## 4.18 source/HexTile.cpp File Reference

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

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



### 4.18.1 Detailed Description

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

A class which defines a tile of a hex map.

## 4.19 source/main.cpp File Reference

Implementation file for [main()](#) for Road To Zero.

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



### Functions

- void [loadAssets](#) ([AssetsManager](#) ∗assets_manager_ptr)

  *Helper function to load game assets.*
- sf::RenderWindow ∗ [constructRenderWindow](#) (void)

  *Helper function to construct render window.*
- int [main](#) (int argc, char ∗∗argv)

### 4.19.1 Detailed Description

Implementation file for main() for Road To Zero.

### 4.19.2 Function Documentation

#### 4.19.2.1 constructRenderWindow()

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

Helper function to construct render window.

**Returns**

> Pointer to the render window.

```
80 {
81     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
82         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
83         "Road To Zero"
84     );
85
86     return render_window_ptr;
87 }   /* constructRenderWindow() */
```

#### 4.19.2.2 loadAssets()

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

Helper function to load game assets.

**Parameters**

| assets_manager_ptr | Pointer to the assets manager. |
|---|---|

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

```
51        );
52
53        assets_manager_ptr->loadTexture(
54            "assets/tile_sheets/water_waves_64x64_1.png",
55            "water_waves_64x64_1"
56        );
57
58        assets_manager_ptr->loadTexture(
59            "assets/tile_sheets/water_shimmer_64x64_1.png",
60            "water_shimmer_64x64_1"
61        );
62
63        return;
64 }    /* loadAssets() */
```

#### 4.19.2.3 main()

```
int main (
            int argc,
            char ** argv )
96 {
97     // 1. load assets
98     AssetsManager assets_manager;
99     loadAssets(&assets_manager);
100
101    // 2. construct render window
102    sf::RenderWindow* render_window_ptr = constructRenderWindow();
103
104    // 3. start game loop
105    bool quit_game = false;
106
107    while (not quit_game) {
108        Game game(render_window_ptr, &assets_manager);
109        quit_game = game.run();
110    }
111
112    // 4. clean up
113    render_window_ptr->close();
114    delete render_window_ptr;
115
116    return 0;
117 }    /* main() */
```

## 4.20 source/TileImprovement.cpp File Reference

Implementation file for the TileImprovement class.

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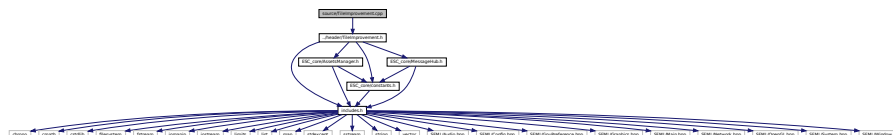


### 4.20.1 Detailed Description

Implementation file for the TileImprovement class.

A base class for the tile improvement hierarchy.

# Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL https://www.sfml-dev.org/. 112

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 110

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 69, 97

# Index