Road To Zero - The Microgrid Management Game

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1   Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AssetsManager Class Reference

A class which manages visual and sound assets.

```
#include <AssetsManager.h>
```

Public Member Functions

- AssetsManager (void)

  Constructor for the AssetsManager class.
- void loadFont (std::string, std::string)

  Method to load a font and insert it into the font map.
- void loadTexture (std::string, std::string)

  Method to load a texture and insert it into the texture map.
- void loadSound (std::string, std::string)

  Method to load a sound and insert it into the sound map. Automatically creates a corresponding sf::SoundBuffer.
- void loadTrack (std::string, std::string)

  Method to load a track (sf::Music) and insert it into the track map.
- sf::Font ∗ getFont (std::string)

  Method to get font associated with given font key.
- sf::Texture ∗ getTexture (std::string)

  Method to get texture associated with given texture key.
- sf::SoundBuffer ∗ getSoundBuffer (std::string)

  Method to get soundbuffer associated with given sound key.
- sf::Sound ∗ getSound (std::string)

  Method to get sound associated with given sound key.
- void playTrack (void)

  Method to play the current track.
- void pauseTrack (void)

  Method to pause the current track.
- void stopTrack (void)

  Method to stop the current track.
- void nextTrack (void)

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

- void previousTrack (void)

  Method to return to the previous track. Wraps around if the beginning of the track map is reached.
- std::string getCurrentTrackKey (void)

  Method to get track key for current track.
- sf::SoundSource::Status getTrackStatus (void)

  Method to get the status of the current track.
- void clear (void)

  Method to clear all loaded assets.
- ∼AssetsManager (void)

  Destructor for the AssetsManager class.

Public Attributes

- std::map< std::string, sf::Font ∗ > font_map

  A map of pointers to loaded fonts.
- std::map< std::string, sf::Texture ∗ > texture_map

  A map of pointers to loaded textures.
- std::map< std::string, sf::SoundBuffer ∗ > soundbuffer_map

  A map of pointers to sound buffers.
- std::map< std::string, sf::Sound ∗ > sound_map

  A map of pointers to loaded sounds.
- std::map< std::string, sf::Music ∗ >::iterator current_track

  A map iterator which corresponds to the current track (i.e., the track currently being played).
- std::map< std::string, sf::Music ∗ > track_map

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

Private Member Functions

- void __loadSoundBuffer (std::string, std::string)

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

4.1.1 Detailed Description

A class which manages visual and sound assets.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AssetsManager()

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

Constructor for the AssetsManager class.

```
142 {
143     //...
144
145     std::cout « "AssetsManager constructed at " « this « std::endl;
146
147     return;
148 }  /* AssetsManager() */
```

4.1.2.2 ∼AssetsManager()

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

Destructor for the AssetsManager class.

```
771 {
772     this->clear();
773
774     std::cout « "AssetsManager at " « this « " destroyed" « std::endl;
775
776     return;
777 } /* ~AssetsManager() */
```

4.1.3 Member Function Documentation

4.1.3.1 __loadSoundBuffer()

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

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

Parameters

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).

```
79 {
80     //  1. check key, throw error if already in use
81     if (this->soundbuffer_map.count(sound_key) > 0) {
82         std::string error_str = "ERROR  AssetsManager::__loadSoundBuffer()  sound key ";
83         error_str += sound_key;
84         error_str += " is already in use";
85
86         this->clear();
87
88         #ifdef _WIN32
89             std::cout « error_str « std::endl;
90         #endif  /* _WIN32 */
91
92         throw std::runtime_error(error_str);
93     }
94
95
96     //  2. load from file, throw error on fail
97     sf::SoundBuffer* soundbuffer_ptr = new sf::SoundBuffer();
98
99     if (not soundbuffer_ptr->loadFromFile(path_2_sound)) {
100         std::string error_str = "ERROR  AssetsManager::__loadSoundBuffer()  could not load ";
101         error_str += "soundbuffer at ";
102         error_str += path_2_sound;
103
104         this->clear();
105
106         #ifdef _WIN32
107             std::cout « error_str « std::endl;
108         #endif  /* _WIN32 */
109
110         throw std::runtime_error(error_str);
111     }
112
113
```

```
114    //  3. insert into soundbuffer map
115    this->soundbuffer_map.insert(
116        std::pair<std::string, sf::SoundBuffer*>(sound_key, soundbuffer_ptr)
117    );
118
119    std::cout « "SoundBuffer " « sound_key « " inserted into soundbuffer map" «
120        std::endl;
121
122    return;
123 } /* __loadSoundBuffer() */
```

4.1.3.2 clear()

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

Method to clear all loaded assets.
```
678 {
679    //  1. clear fonts
680    std::map<std::string, sf::Font*>::iterator font_iter;
681    for (
682        font_iter = this->font_map.begin();
683        font_iter != this->font_map.end();
684        font_iter++
685    ) {
686        delete font_iter->second;
687
688        std::cout « "Font " « font_iter->first « " deleted from font map" «
689            std::endl;
690    }
691    this->font_map.clear();
692
693
694    //  2. clear textures
695    std::map<std::string, sf::Texture*>::iterator texture_iter;
696    for (
697        texture_iter = this->texture_map.begin();
698        texture_iter != this->texture_map.end();
699        texture_iter++
700    ) {
701        delete texture_iter->second;
702
703        std::cout « "Texture " « texture_iter->first « " deleted from texture map" «
704            std::endl;
705    }
706    this->texture_map.clear();
707
708
709    //  3. clear sound buffers
710    std::map<std::string, sf::SoundBuffer*>::iterator soundbuffer_iter;
711    for (
712        soundbuffer_iter = this->soundbuffer_map.begin();
713        soundbuffer_iter != this->soundbuffer_map.end();
714        soundbuffer_iter++
715    ) {
716        delete soundbuffer_iter->second;
717
718        std::cout « "SoundBuffer " « soundbuffer_iter->first «
719            " deleted from soundbuffer map" « std::endl;
720    }
721    this->soundbuffer_map.clear();
722
723
724    //  4. clear sounds
725    std::map<std::string, sf::Sound*>::iterator sound_iter;
726    for (
727        sound_iter = this->sound_map.begin();
728        sound_iter != this->sound_map.end();
729        sound_iter++
730    ) {
731        sound_iter->second->stop();
732        delete sound_iter->second;
733
734        std::cout « "Sound " « sound_iter->first « " deleted from sound map" «
735            std::endl;
736    }
737    this->sound_map.clear();
738
```

```
739
740     //  5. clear tracks
741     std::map<std::string, sf::Music*>::iterator track_iter;
742     for (
743         track_iter = this->track_map.begin();
744         track_iter != this->track_map.end();
745         track_iter++
746     ) {
747         track_iter->second->stop();
748         delete track_iter->second;
749
750         std::cout « "Track " « track_iter->first « " deleted from track map" «
751             std::endl;
752     }
753     this->track_map.clear();
754
755     return;
756 }   /* clear() */
```

4.1.3.3 getCurrentTrackKey()

```
std::string AssetsManager::getCurrentTrackKey (
            void  )
```

Method to get track key for current track.

Returns

The track key for the current track.

```
642 {
643     return this->current_track->first;
644 }   /* getCurrentTrackKey() */
```

4.1.3.4 getFont()

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

Method to get font associated with given font key.

Parameters

font_key	A key associated with the font (for indexing into the font map).

Returns

A pointer to the corresponding font.

```
383 {
384     //  1. check key, throw error if not found
385     if (this->font_map.count(font_key) <= 0) {
386         std::string error_str = "ERROR  AssetsManager::getFont()  font key ";
387         error_str += font_key;
388         error_str += " is not contained in font map";
389
390         this->clear();
391
392         #ifdef _WIN32
```

```
393              std::cout « error_str « std::endl;
394          #endif  /* _WIN32 */
395
396          throw std::runtime_error(error_str);
397      }
398
399      return this->font_map[font_key];
400 }   /* getFont() */
```

4.1.3.5 getSound()

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

Method to get sound associated with given sound key.

Parameters

sound_key	A key associated with the sound (for indexing into the sound map).

Returns

A pointer to the corresponding sound.

```
493 {
494     //  1. check key, throw error if not found
495     if (this->sound_map.count(sound_key) <= 0) {
496         std::string error_str = "ERROR  AssetsManager::getSound()  sound key ";
497         error_str += sound_key;
498         error_str += " is not contained in sound map";
499
500         this->clear();
501
502         #ifdef _WIN32
503             std::cout « error_str « std::endl;
504         #endif  /* _WIN32 */
505
506         throw std::runtime_error(error_str);
507     }
508
509     return this->sound_map[sound_key];
510 }   /* getSound() */
```

4.1.3.6 getSoundBuffer()

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

Method to get soundbuffer associated with given sound key.

Parameters

sound_key	A key associated with the soundbuffer (for indexing into the soundbuffer map).

Returns

A pointer to the corresponding soundbuffer.

```
457 {
458     //  1. check key, throw error if not found
459     if (this->soundbuffer_map.count(sound_key) <= 0) {
460         std::string error_str = "ERROR  AssetsManager::getSoundBuffer()  sound key ";
461         error_str += sound_key;
462         error_str += " is not contained in soundbuffer map";
463
464         this->clear();
465
466         #ifdef _WIN32
467             std::cout << error_str << std::endl;
468         #endif  /* _WIN32 */
469
470         throw std::runtime_error(error_str);
471     }
472
473     return this->soundbuffer_map[sound_key];
474 }   /* getSoundBuffer() */
```

4.1.3.7 getTexture()

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

Method to get texture associated with given texture key.

Parameters

texture_key	A key associated with the texture (for indexing into the texture map).

Returns

A pointer to the corresponding texture.

```
420 {
421     //  1. check key, throw error if not found
422     if (this->texture_map.count(texture_key) <= 0) {
423         std::string error_str = "ERROR  AssetsManager::getTexture()  texture key ";
424         error_str += texture_key;
425         error_str += " is not contained in texture map";
426
427         this->clear();
428
429         #ifdef _WIN32
430             std::cout << error_str << std::endl;
431         #endif  /* _WIN32 */
432
433         throw std::runtime_error(error_str);
434     }
435
436     return this->texture_map[texture_key];
437 }   /* getTexture() */
```

4.1.3.8 getTrackStatus()

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

Method to get the status of the current track.

Returns

The status of the current track.

```
661 {
662     return this->current_track->second->getStatus();
663 }  /* getTrackStatus */
```

4.1.3.9 loadFont()

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

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

Parameters

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).

```
167 {
168     //  1. check key, throw error if already in use
169     if (this->font_map.count(font_key) > 0) {
170         std::string error_str = "ERROR  AssetsManager::loadFont()  font key ";
171         error_str += font_key;
172         error_str += " is already in use";
173
174         this->clear();
175
176         #ifdef _WIN32
177             std::cout « error_str « std::endl;
178         #endif  /* _WIN32 */
179
180         throw std::runtime_error(error_str);
181     }
182
183
184     //  2. load from file, throw error on fail
185     sf::Font* font_ptr = new sf::Font();
186
187     if (not font_ptr->loadFromFile(path_2_font)) {
188         std::string error_str = "ERROR  AssetsManager::loadFont()  could not load ";
189         error_str += "font at ";
190         error_str += path_2_font;
191
192         this->clear();
193
194         #ifdef _WIN32
195             std::cout « error_str « std::endl;
196         #endif  /* _WIN32 */
197
198         throw std::runtime_error(error_str);
199     }
200
201
202     //  3. insert into font map
203     this->font_map.insert(std::pair<std::string, sf::Font*>(font_key, font_ptr));
204
205     std::cout « "Font " « font_key « " inserted into font map" « std::endl;
206
207     return;
208 }  /* loadFont() */
```

4.1.3.10 loadSound()

```
void AssetsManager::loadSound (
```

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

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

Parameters

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).

```
291 {
292     //  1. create an associated sf::SoundBuffer
293     this->__loadSoundBuffer(path_2_sound, sound_key);
294
295     //  2. associate sf::Sound with sf::SoundBuffer
296     sf::Sound* sound_ptr = new sf::Sound();
297     sound_ptr->setBuffer(*(this->soundbuffer_map[sound_key]));
298
299     //  3. insert into sound map
300     this->sound_map.insert(std::pair<std::string, sf::Sound*>(sound_key, sound_ptr));
301
302     std::cout « "Sound " « sound_key « " inserted into sound map" « std::endl;
303
304     return;
305 }   /* loadSound() */
```

4.1.3.11 loadTexture()

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

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

Parameters

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).

```
228 {
229     //  1. check key, throw error if already in use
230     if (this->texture_map.count(texture_key) > 0) {
231         std::string error_str = "ERROR  AssetsManager::loadTexture()  texture key ";
232         error_str += texture_key;
233         error_str += " is already in use";
234
235         this->clear();
236
237         #ifdef _WIN32
238             std::cout « error_str « std::endl;
239         #endif  /* _WIN32 */
240
241         throw std::runtime_error(error_str);
242     }
243
244
245     //  2. load from file, throw error on fail
246     sf::Texture* texture_ptr = new sf::Texture();
247
248     if (not texture_ptr->loadFromFile(path_2_texture)) {
249         std::string error_str = "ERROR  AssetsManager::loadTexture()  could not load ";
250         error_str += "texture at ";
251         error_str += path_2_texture;
252
253         this->clear();
254
255         #ifdef _WIN32
256             std::cout « error_str « std::endl;
```

```
257        #endif  /* _WIN32 */
258
259        throw std::runtime_error(error_str);
260    }
261
262
263    //  3. insert into texture map
264    this->texture_map.insert(
265        std::pair<std::string, sf::Texture*>(texture_key, texture_ptr)
266    );
267
268    std::cout << "Texture " << texture_key << " inserted into texture map" << std::endl;
269
270    return;
271 }   /* loadTexture() */
```

4.1.3.12 loadTrack()

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

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

Parameters

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).

```
324 {
325    //  1. check key, throw error if already in use
326    if (this->track_map.count(track_key) > 0) {
327        std::string error_str = "ERROR  AssetsManager::loadTrack()  track key ";
328        error_str += track_key;
329        error_str += " is already in use";
330
331        this->clear();
332
333        #ifdef _WIN32
334            std::cout << error_str << std::endl;
335        #endif  /* _WIN32 */
336
337        throw std::runtime_error(error_str);
338    }
339
340    //  2. open from file, throw error on fail
341    sf::Music* track_ptr = new sf::Music();
342
343    if (not track_ptr->openFromFile(path_2_track)) {
344        std::string error_str = "ERROR  AssetsManager::loadTrack()  could not open ";
345        error_str += "track at ";
346        error_str += path_2_track;
347
348        this->clear();
349
350        #ifdef _WIN32
351            std::cout << error_str << std::endl;
352        #endif  /* _WIN32 */
353
354        throw std::runtime_error(error_str);
355    }
356
357    //  3. insert into track map
358    this->track_map.insert(std::pair<std::string, sf::Music*>(track_key, track_ptr));
359    this->current_track = this->track_map.begin();
360
361    std::cout << "Track " << track_key << " inserted into track map" << std::endl;
362
363    return;
364 }   /* loadTrack() */
```

4.1.3.13 nextTrack()

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

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

```
583 {
584     //  1. stop current track
585     this->stopTrack();
586
587     //  2. increment current track
588     this->current_track++;
589
590     //  3. handle wrap around
591     if (this->current_track == this->track_map.end()) {
592         this->current_track = this->track_map.begin();
593     }
594
595     return;
596 }   /* nextTrack() */
```

4.1.3.14 pauseTrack()

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

Method to pause the current track.

```
544 {
545     this->current_track->second->pause();
546
547     return;
548 }   /* pauseTrack() */
```

4.1.3.15 playTrack()

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

Method to play the current track.

```
525 {
526     this->current_track->second->play();
527
528     return;
529 }   /* playTrack() */
```

4.1.3.16 previousTrack()

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

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

```
612 {
613     //  1. stop current track
614     this->stopTrack();
615
616     //  2. handle wrap around
617     if (this->current_track == this->track_map.begin()) {
618         this->current_track = this->track_map.end();
619     }
620
621     //  3. decrement current track
622     this->current_track--;
623
624     return;
625 }   /* previousTrack() */
```

4.1.3.17 stopTrack()

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

Method to stop the current track.

```
563 {
564     this->current_track->second->stop();
565
566     return;
567 } /* stopTrack() */
```

4.1.4 Member Data Documentation

4.1.4.1 current_track

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

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

4.1.4.2 font_map

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

A map of pointers to loaded fonts.

4.1.4.3 sound_map

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

A map of pointers to loaded sounds.

4.1.4.4 soundbuffer_map

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

A map of pointers to sound buffers.

4.1.4.5 texture_map

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

A map of pointers to loaded textures.

4.1.4.6 track_map

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

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

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

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

4.2 ContextMenu Class Reference

A class which defines a context menu for the game.

`#include <ContextMenu.h>`

Collaboration diagram for ContextMenu:
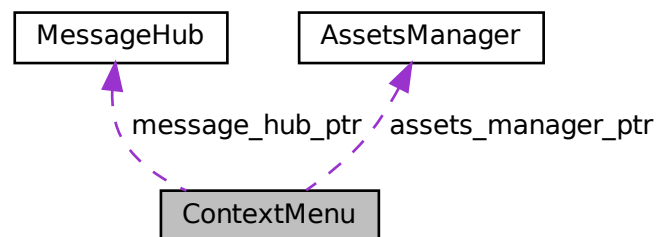
Public Member Functions

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

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

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

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

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

    Destructor for the ContextMenu class.

Public Attributes

- ConsoleState console_state

    The current state of the console screen.
- bool console_string_changed

    Boolean which indicates if console string just changed.
- bool game_menu_up

    Indicates whether or not the game menu is up.
- size_t console_substring_idx

    The current final index of the console string draw.
- unsigned long long int frame

    The current frame of this object.
- double position_x

    The position of the object.
- double position_y

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

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

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

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

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

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

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

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

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

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

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

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

    The right framing of the console screen.

Private Member Functions

- void __setUpMenuFrame (void)

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

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

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

Helper method to draw visual screen frame.

- void __setUpConsoleScreen (void)

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

- void __setUpConsoleScreenFrame (void)

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

- void __drawConsoleScreenFrame (void)

    Helper method to draw console screen frame.

- void __setConsoleState (ConsoleState)

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

- void __setConsoleString (void)

    Helper method to set console string depending on console state.

- void __drawConsoleText (void)

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

- void __handleKeyPressEvents (void)

    Helper method to handle key press events.

- void __handleMouseButtonEvents (void)

    Helper method to handle mouse button events.

- void __sendQuitGameMessage (void)

    Helper method to format and send a quit game message.

- void __sendRestartGameMessage (void)

    Helper method to format and send a restart game message.

Private Attributes

- sf::Event ∗ event_ptr

    A pointer to the event class.

- sf::RenderWindow ∗ render_window_ptr

    A pointer to the render window.

- AssetsManager ∗ assets_manager_ptr

    A pointer to the assets manager.

- MessageHub ∗ message_hub_ptr

    A pointer to the message hub.

4.2.1 Detailed Description

A class which defines a context menu for the game.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 ContextMenu()

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

Constructor for the ContextMenu class.

Parameters

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.

```
849 {
850     //  1. set attributes
851
852     //  1.1. private
853     this->event_ptr = event_ptr;
854     this->render_window_ptr = render_window_ptr;
855
856     this->assets_manager_ptr = assets_manager_ptr;
857     this->message_hub_ptr = message_hub_ptr;
858
859     //  1.2. public
860     this->console_state = ConsoleState :: NONE_STATE;
861     this->__setConsoleState(ConsoleState :: READY);
862
863     this->console_string_changed = true;
864     this->game_menu_up = false;
865
866     this->frame = 0;
867
868     this->position_x = GAME_WIDTH;
869     this->position_y = 0;
870
871     //  2. set up and position drawable attributes
872     this->__setUpMenuFrame();
873     this->__setUpVisualScreen();
874     this->__setUpVisualScreenFrame();
875     this->__setUpConsoleScreen();
876     this->__setUpConsoleScreenFrame();
877
878     std::cout « "ContextMenu constructed at " « this « std::endl;
879
880     return;
881 }   /* ContextMenu() */
```

4.2.2.2 ~ContextMenu()

```
ContextMenu::~ContextMenu (
            void  )
```

Destructor for the ContextMenu class.
```
1031 {
1032     std::cout « "ContextMenu at " « this « " destroyed" « std::endl;
1033
1034     return;
1035 }   /* ~ContextMenu() */
```

4.2.3 Member Function Documentation

4.2.3.1 __drawConsoleScreenFrame()

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

Helper method to draw console screen frame.

```
467 {
468     this->render_window_ptr->draw(this->console_screen_frame_top);
469     this->render_window_ptr->draw(this->console_screen_frame_left);
470     this->render_window_ptr->draw(this->console_screen_frame_bottom);
471     this->render_window_ptr->draw(this->console_screen_frame_right);
472
473     return;
474 } /* __drawContextScreenFrame() */
```

4.2.3.2 __drawConsoleText()

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

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

```
590 {
591     //  1. set up console text (drawable)
592     sf::Text console_text;
593
594     if (this->console_string_changed) {
595         this->assets_manager_ptr->getSound("console string print")->play();
596
597         console_text.setString(this->console_string.substr(0, this->console_substring_idx));
598
599         this->console_substring_idx++;
600
601         while (
602             (this->console_string.substr(0, this->console_substring_idx).back() == ' ') or
603             (this->console_string.substr(0, this->console_substring_idx).back() == '\n')
604         ) {
605             this->console_substring_idx++;
606
607             if (this->console_substring_idx >= this->console_string.size()) {
608                 break;
609             }
610         }
611
612         if (this->console_substring_idx >= this->console_string.size()) {
613             this->console_string_changed = false;
614         }
615     }
616
617     else {
618         console_text.setString(this->console_string);
619     }
620
621     console_text.setFont(*(this->assets_manager_ptr->getFont("Glass_TTY_VT220")));
622     console_text.setCharacterSize(16);
623     console_text.setFillColor(MONOCHROME_TEXT_GREEN);
624
625     console_text.setPosition(
626         this->position_x - 50 - 300 + 16,
627         this->position_y + GAME_HEIGHT - 50 - 340 + 16
628     );
629
630
631     //  2. draw console text
632     this->render_window_ptr->draw(console_text);
633
634
635     //  3. assemble and draw blinking console cursor
636     if ((this->frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
637         sf::RectangleShape console_cursor(sf::Vector2f(10, 16));
638
639         console_cursor.setFillColor(MONOCHROME_TEXT_GREEN);
640
641         console_cursor.setPosition(
642             console_text.getPosition().x,
643             console_text.getPosition().y + console_text.getLocalBounds().height + 10
644         );
645
646         this->render_window_ptr->draw(console_cursor);
647     }
648
649     //  4. updating frame count if console is in menu state
650     if (this->console_state == ConsoleState :: MENU) {
651         std::string frame_count_string = "FRAME: ";
652         frame_count_string += std::to_string(this->frame);
```

```
653
654        sf::Text frame_count_text(
655            frame_count_string,
656            *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
657            16
658        );
659
660        frame_count_text.setFillColor(MONOCHROME_TEXT_GREEN);
661
662        frame_count_text.setPosition(
663            console_text.getPosition().x,
664            console_text.getPosition().y + console_text.getLocalBounds().height - 10
665        );
666
667        this->render_window_ptr->draw(frame_count_text);
668    }
669
670    return;
671 }   /* __drawConsoleText() */
```

4.2.3.3 __drawVisualScreenFrame()

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

Helper method to draw visual screen frame.

```
242 {
243    this->render_window_ptr->draw(this->visual_screen_frame_top);
244    this->render_window_ptr->draw(this->visual_screen_frame_left);
245    this->render_window_ptr->draw(this->visual_screen_frame_bottom);
246    this->render_window_ptr->draw(this->visual_screen_frame_right);
247
248    return;
249 }   /* __drawVisualScreenFrame() */
```

4.2.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
686 {
687    switch (this->event_ptr->key.code) {
688        case (sf::Keyboard::Escape): {
689            if (this->console_state == ConsoleState :: MENU) {
690                this->__setConsoleState(ConsoleState :: READY);
691            }
692
693            else {
694                this->__setConsoleState(ConsoleState :: MENU);
695            }
696
697            break;
698        }
699
700
701        case (sf::Keyboard::Q): {
702            if (this->console_state == ConsoleState :: MENU) {
703                this->__sendQuitGameMessage();
704            }
705        }
706
707
708        case (sf::Keyboard::R): {
709            if (this->console_state == ConsoleState :: MENU) {
710                this->__sendRestartGameMessage();
711            }
712        }
713
```

```
714
715         default: {
716             // do nothing!
717
718             break;
719         }
720     }
721
722     return;
723 } /* __handleKeyPressEvents() */
```

4.2.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
738 {
739     switch (this->event_ptr->mouseButton.button) {
740         case (sf::Mouse::Left): {
741             //...
742
743             break;
744         }
745
746
747         case (sf::Mouse::Right): {
748             //...
749
750             break;
751         }
752
753
754         default: {
755             // do nothing!
756
757             break;
758         }
759     }
760
761     return;
762 } /* __handleMouseButtonEvents() */
```

4.2.3.6 __sendQuitGameMessage()

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

Helper method to format and send a quit game message.

```
777 {
778     Message quit_game_message;
779
780     quit_game_message.channel = GAME_CHANNEL;
781     quit_game_message.subject = "quit game";
782
783     this->message_hub_ptr->sendMessage(quit_game_message);
784
785     std::cout << "Quit game message sent by " << this << std::endl;
786     return;
787 } /* __sendQuitGameMessage() */
```

4.2.3.7 __sendRestartGameMessage()

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

Helper method to format and send a restart game message.

```
802 {
803     Message restart_game_message;
804
805     restart_game_message.channel = GAME_CHANNEL;
806     restart_game_message.subject = "restart game";
807
808     this->message_hub_ptr->sendMessage(restart_game_message);
809
810     std::cout « "Restart game message sent by " « this « std::endl;
811     return;
812 }   /* __sendRestartGameMessage() */
```

4.2.3.8 __setConsoleState()

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

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

Parameters

console_state	The state (ConsoleState) to set the console to.

```
491 {
492     //  1. if no change, do nothing
493     if (this->console_state == console_state) {
494         return;
495     }
496
497     //  2. update console state, set console string accordingly
498     this->console_state = console_state;
499     this->__setConsoleString();
500
501     return;
502 }   /* __setConsoleState() */
```

4.2.3.9 __setConsoleString()

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

Helper method to set console string depending on console state.

```
517 {
518     this->console_string_changed = true;
519     this->console_substring_idx = 0;
520
521     this->console_string.clear();
522
523     switch (this->console_state) {
524         case (ConsoleState :: MENU): {
525             //          32 char x 17 line console "--------------------------------\n";
526             this->console_string                 = "        **** MENU ****         \n";
527             this->console_string                += "                               \n";
528             this->console_string                += "[R]:  RESTART                  \n";
529             this->console_string                += "                               \n";
530             this->console_string                += "[TAB]:  TOGGLE RESOURCE OVERLAY \n";
```

```
531            this->console_string                  += "[T]:   TUTORIAL                \n";
532            this->console_string                  += "                                \n";
533            this->console_string                  += "                                \n";
534            this->console_string                  += "                                \n";
535            this->console_string                  += "                                \n";
536            this->console_string                  += "                                \n";
537            this->console_string                  += "                                \n";
538            this->console_string                  += "                                \n";
539            this->console_string                  += "[Q]:    QUIT                    \n";
540            this->console_string                  += "[ESC]:  CLOSE MENU              \n";
541            this->console_string                  += "                                \n";
542
543            break;
544        }
545
546
547        case (ConsoleState :: TILE): {
548            // take console string from tile state message
549
550            break;
551        }
552
553
554        default: {
555            //          32 char x 17 line console "--------------------------------\n";
556            this->console_string                   = "  **** RTZ 64 CONTEXT V12 ****  \n";
557            this->console_string                  += "                                \n";
558            this->console_string                  += "64K RAM SYSTEM  38911 BYTES FREE\n";
559            this->console_string                  += "                                \n";
560            this->console_string                  += "[TAB]:  TOGGLE RESOURCE OVERLAY \n";
561            this->console_string                  += "                                \n";
562            this->console_string                  += "[ESC]:       MENU               \n";
563            this->console_string                  += "[LEFT CLICK]:  TILE INFO/OPTIONS\n";
564            this->console_string                  += "[RIGHT CLICK]: CLEAR SELECTION  \n";
565            this->console_string                  += "                                \n";
566            this->console_string                  += "[ENTER]:  END TURN              \n";
567            this->console_string                  += "                                \n";
568            this->console_string                  += "READY.                          ";
569
570            break;
571        }
572    }
573
574    return;
575 }  /* __setConsoleString() */
```

4.2.3.10 __setUpConsoleScreen()

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

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

```
264 {
265    this->console_screen.setSize(sf::Vector2f(300, 340));
266    this->console_screen.setOrigin(300, 340);
267    this->console_screen.setPosition(
268        this->position_x - 50,
269        this->position_y + GAME_HEIGHT - 50
270    );
271    this->console_screen.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
272
273    return;
274 }  /* __setUpConsoleScreen() */
```

4.2.3.11 __setUpConsoleScreenFrame()

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

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

```
289 {
290     int n_points = 4;
291
292     //  1. top framing
293     this->console_screen_frame_top.setPointCount(n_points);
294
295     this->console_screen_frame_top.setPoint(
296         0,
297         sf::Vector2f(
298             this->position_x - 50,
299             this->position_y + GAME_HEIGHT - 50 - 340
300         )
301     );
302     this->console_screen_frame_top.setPoint(
303         1,
304         sf::Vector2f(
305             this->position_x - 50 + 16,
306             this->position_y + GAME_HEIGHT - 50 - 340 - 16
307         )
308     );
309     this->console_screen_frame_top.setPoint(
310         2,
311         sf::Vector2f(
312             this->position_x - 350 - 16,
313             this->position_y + GAME_HEIGHT - 50 - 340 - 16
314         )
315     );
316     this->console_screen_frame_top.setPoint(
317         3,
318         sf::Vector2f(
319             this->position_x - 350,
320             this->position_y + GAME_HEIGHT - 50 - 340
321         )
322     );
323
324     this->console_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
325
326     this->console_screen_frame_top.setOutlineThickness(2);
327     this->console_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
328
329     this->console_screen_frame_top.move(0, -2);
330
331
332     //  2. left framing
333     this->console_screen_frame_left.setPointCount(n_points);
334
335     this->console_screen_frame_left.setPoint(
336         0,
337         sf::Vector2f(
338             this->position_x - 350,
339             this->position_y + GAME_HEIGHT - 50 - 340
340         )
341     );
342     this->console_screen_frame_left.setPoint(
343         1,
344         sf::Vector2f(
345             this->position_x - 350 - 16,
346             this->position_y + GAME_HEIGHT - 50 - 340 - 16
347         )
348     );
349     this->console_screen_frame_left.setPoint(
350         2,
351         sf::Vector2f(
352             this->position_x - 350 - 16,
353             this->position_y + GAME_HEIGHT - 50 + 16
354         )
355     );
356     this->console_screen_frame_left.setPoint(
357         3,
358         sf::Vector2f(
359             this->position_x - 350,
360             this->position_y + GAME_HEIGHT - 50
361         )
362     );
363
364     this->console_screen_frame_left.setFillColor(VISUAL_SCREEN_FRAME_GREY);
365
366     this->console_screen_frame_left.setOutlineThickness(2);
367     this->console_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
368
369     this->console_screen_frame_left.move(-2, 0);
370
371
372     //  3. bottom framing
373     this->console_screen_frame_bottom.setPointCount(n_points);
374
```

```
375      this->console_screen_frame_bottom.setPoint(
376          0,
377          sf::Vector2f(
378              this->position_x - 350,
379              this->position_y + GAME_HEIGHT - 50
380          )
381      );
382      this->console_screen_frame_bottom.setPoint(
383          1,
384          sf::Vector2f(
385              this->position_x - 350 - 16,
386              this->position_y + GAME_HEIGHT - 50 + 16
387          )
388      );
389      this->console_screen_frame_bottom.setPoint(
390          2,
391          sf::Vector2f(
392              this->position_x - 50 + 16,
393              this->position_y + GAME_HEIGHT - 50 + 16
394          )
395      );
396      this->console_screen_frame_bottom.setPoint(
397          3,
398          sf::Vector2f(
399              this->position_x - 50,
400              this->position_y + GAME_HEIGHT - 50
401          )
402      );
403
404      this->console_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
405
406      this->console_screen_frame_bottom.setOutlineThickness(2);
407      this->console_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
408
409      this->console_screen_frame_bottom.move(0, 2);
410
411
412      //  4. right framing
413      this->console_screen_frame_right.setPointCount(n_points);
414
415      this->console_screen_frame_right.setPoint(
416          0,
417          sf::Vector2f(
418              this->position_x - 50,
419              this->position_y + GAME_HEIGHT - 50
420          )
421      );
422      this->console_screen_frame_right.setPoint(
423          1,
424          sf::Vector2f(
425              this->position_x - 50 + 16,
426              this->position_y + GAME_HEIGHT - 50 + 16
427          )
428      );
429      this->console_screen_frame_right.setPoint(
430          2,
431          sf::Vector2f(
432              this->position_x - 50 + 16,
433              this->position_y + GAME_HEIGHT - 50 - 340 - 16
434          )
435      );
436      this->console_screen_frame_right.setPoint(
437          3,
438          sf::Vector2f(
439              this->position_x - 50,
440              this->position_y + GAME_HEIGHT - 50 - 340
441          )
442      );
443
444      this->console_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
445
446      this->console_screen_frame_right.setOutlineThickness(2);
447      this->console_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
448
449      this->console_screen_frame_right.move(2, 0);
450
451      return;
452 }   /* __setUpConsoleScreenFrame() */
```

4.2.3.12 __setUpMenuFrame()

```
void ContextMenu::__setUpMenuFrame (
```

```
            void  )  [private]
```

Helper method to set up context menu frame (drawable).
```
68 {
69     this->menu_frame.setSize(sf::Vector2f(400, GAME_HEIGHT));
70     this->menu_frame.setOrigin(400, 0);
71     this->menu_frame.setPosition(this->position_x, this->position_y);
72     this->menu_frame.setFillColor(MENU_FRAME_GREY);
73
74     return;
75 }   /* __setUpMenuFrame() */
```

4.2.3.13   __setUpVisualScreen()

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

Helper method to set up context menu visual screen (drawable).
```
90 {
91     this->visual_screen.setSize(sf::Vector2f(300, 300));
92     this->visual_screen.setOrigin(300, 0);
93     this->visual_screen.setPosition(this->position_x - 50, this->position_y + 50);
94     this->visual_screen.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
95
96     return;
97 }   /* __setUpVisualScreen() */
```

4.2.3.14   __setUpVisualScreenFrame()

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

Helper method to set up framing for context menu visual screen (drawable).
```
112 {
113     int n_points = 4;
114
115     //  1. top framing
116     this->visual_screen_frame_top.setPointCount(n_points);
117
118     this->visual_screen_frame_top.setPoint(
119         0,
120         sf::Vector2f(this->position_x - 50, this->position_y + 50)
121     );
122     this->visual_screen_frame_top.setPoint(
123         1,
124         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
125     );
126     this->visual_screen_frame_top.setPoint(
127         2,
128         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
129     );
130     this->visual_screen_frame_top.setPoint(
131         3,
132         sf::Vector2f(this->position_x - 350, this->position_y + 50)
133     );
134
135     this->visual_screen_frame_top.setFillColor(VISUAL_SCREEN_FRAME_GREY);
136
137     this->visual_screen_frame_top.setOutlineThickness(2);
138     this->visual_screen_frame_top.setOutlineColor(sf::Color(0, 0, 0, 255));
139
140     this->visual_screen_frame_top.move(0, -2);
141
142
143     //  2. left framing
144     this->visual_screen_frame_left.setPointCount(n_points);
145
146     this->visual_screen_frame_left.setPoint(
```

```
147         0,
148         sf::Vector2f(this->position_x - 350, this->position_y + 50)
149     );
150     this->visual_screen_frame_left.setPoint(
151         1,
152         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 50 - 16)
153     );
154     this->visual_screen_frame_left.setPoint(
155         2,
156         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
157     );
158     this->visual_screen_frame_left.setPoint(
159         3,
160         sf::Vector2f(this->position_x - 350, this->position_y + 350)
161     );
162
163     this->visual_screen_frame_left.setFillColor(VISUAL_SCREEN_FRAME_GREY);
164
165     this->visual_screen_frame_left.setOutlineThickness(2);
166     this->visual_screen_frame_left.setOutlineColor(sf::Color(0, 0, 0, 255));
167
168     this->visual_screen_frame_left.move(-2, 0);
169
170
171     //  3. bottom framing
172     this->visual_screen_frame_bottom.setPointCount(n_points);
173
174     this->visual_screen_frame_bottom.setPoint(
175         0,
176         sf::Vector2f(this->position_x - 350, this->position_y + 350)
177     );
178     this->visual_screen_frame_bottom.setPoint(
179         1,
180         sf::Vector2f(this->position_x - 350 - 16, this->position_y + 350 + 16)
181     );
182     this->visual_screen_frame_bottom.setPoint(
183         2,
184         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
185     );
186     this->visual_screen_frame_bottom.setPoint(
187         3,
188         sf::Vector2f(this->position_x - 50, this->position_y + 350)
189     );
190
191     this->visual_screen_frame_bottom.setFillColor(VISUAL_SCREEN_FRAME_GREY);
192
193     this->visual_screen_frame_bottom.setOutlineThickness(2);
194     this->visual_screen_frame_bottom.setOutlineColor(sf::Color(0, 0, 0, 255));
195
196     this->visual_screen_frame_bottom.move(0, 2);
197
198
199     //  4. right framing
200     this->visual_screen_frame_right.setPointCount(n_points);
201
202     this->visual_screen_frame_right.setPoint(
203         0,
204         sf::Vector2f(this->position_x - 50, this->position_y + 350)
205     );
206     this->visual_screen_frame_right.setPoint(
207         1,
208         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 350 + 16)
209     );
210     this->visual_screen_frame_right.setPoint(
211         2,
212         sf::Vector2f(this->position_x - 50 + 16, this->position_y + 50 - 16)
213     );
214     this->visual_screen_frame_right.setPoint(
215         3,
216         sf::Vector2f(this->position_x - 50, this->position_y + 50)
217     );
218
219     this->visual_screen_frame_right.setFillColor(VISUAL_SCREEN_FRAME_GREY);
220
221     this->visual_screen_frame_right.setOutlineThickness(2);
222     this->visual_screen_frame_right.setOutlineColor(sf::Color(0, 0, 0, 255));
223
224     this->visual_screen_frame_right.move(2, 0);
225
226     return;
227 }   /* __setUpVisualScreenFrame() */
```

4.2.3.15 draw()

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

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

```
1001 {
1002     //  1. menu frame
1003     this->render_window_ptr->draw(this->menu_frame);
1004
1005     //  2. visual screen
1006     this->render_window_ptr->draw(this->visual_screen);
1007     this->__drawVisualScreenFrame();
1008
1009     //  3. console screen
1010     this->render_window_ptr->draw(this->console_screen);
1011     this->__drawConsoleScreenFrame();
1012     this->__drawConsoleText();
1013
1014     this->frame++;
1015     return;
1016 }   /* draw() */
```

4.2.3.16 processEvent()

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

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

```
896 {
897     if (this->event_ptr->type == sf::Event::KeyPressed) {
898         this->__handleKeyPressEvents();
899     }
900
901     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
902         this->__handleMouseButtonEvents();
903     }
904
905     return;
906 }   /* processEvent() */
```

4.2.3.17 processMessage()

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

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

```
921 {
922     switch (this->console_state) {
923         case (ConsoleState :: TILE): {
924             //  process no tile selected
925             if (not this->message_hub_ptr->isEmpty(NO_TILE_SELECTED_CHANNEL)) {
926                 Message no_tile_selected_message = this->message_hub_ptr->receiveMessage(
927                     NO_TILE_SELECTED_CHANNEL
928                 );
929
930                 if (no_tile_selected_message.subject == "no tile selected") {
931                     this->__setConsoleState(ConsoleState :: READY);
932
933                     std::cout << "No tile selected message received by " << this <<
934                         std::endl;
935                     this->message_hub_ptr->popMessage(NO_TILE_SELECTED_CHANNEL);
936                 }
937             }
938
939             //  process tile state
```

```
940                if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
941                    Message tile_state_message = this->message_hub_ptr->receiveMessage(
942                        TILE_STATE_CHANNEL
943                    );
944
945                    if (tile_state_message.subject == "tile state") {
946                        this->console_string = tile_state_message.string_payload["console string"];
947
948                        this->console_string_changed = true;
949                        this->console_substring_idx = 0;
950
951                        std::cout << "Tile state message received by " << this << std::endl;
952                        this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
953                    }
954                }
955
956                //  process tile selected (subsequent left clicks causing program to hang)
957                if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
958                    this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
959                }
960
961                break;
962            }
963
964        default: {
965            //  process tile selected
966            if (not this->message_hub_ptr->isEmpty(TILE_SELECTED_CHANNEL)) {
967                Message tile_selected_message = this->message_hub_ptr->receiveMessage(
968                    TILE_SELECTED_CHANNEL
969                );
970
971                if (tile_selected_message.subject == "tile selected") {
972                    this->__setConsoleState(ConsoleState :: TILE);
973
974                    std::cout << "Tile selected message received by " << this <<
975                        std::endl;
976                    this->message_hub_ptr->popMessage(TILE_SELECTED_CHANNEL);
977                }
978            }
979
980            break;
981        }
982    }
983
984    return;
985 }   /* processMessage() */
```

4.2.4 Member Data Documentation

4.2.4.1 assets_manager_ptr

AssetsManager* ContextMenu::assets_manager_ptr [private]

A pointer to the assets manager.

4.2.4.2 console_screen

sf::RectangleShape ContextMenu::console_screen

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

4.2.4.3 console_screen_frame_bottom

`sf::ConvexShape ContextMenu::console_screen_frame_bottom`

The bottom framing of the console screen.

4.2.4.4 console_screen_frame_left

`sf::ConvexShape ContextMenu::console_screen_frame_left`

The left framing of the console screen.

4.2.4.5 console_screen_frame_right

`sf::ConvexShape ContextMenu::console_screen_frame_right`

The right framing of the console screen.

4.2.4.6 console_screen_frame_top

`sf::ConvexShape ContextMenu::console_screen_frame_top`

The top framing of the console screen.

4.2.4.7 console_state

[ConsoleState](#) `ContextMenu::console_state`

The current state of the console screen.

4.2.4.8 console_string

`std::string ContextMenu::console_string`

The string to be printed to the console screen.

4.2.4.9 console_string_changed

`bool ContextMenu::console_string_changed`

Boolean which indicates if console string just changed.

4.2.4.10 console_substring_idx

`size_t ContextMenu::console_substring_idx`

The current final index of the console string draw.

4.2.4.11 event_ptr

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

A pointer to the event class.

4.2.4.12 frame

`unsigned long long int ContextMenu::frame`

The current frame of this object.

4.2.4.13 game_menu_up

`bool ContextMenu::game_menu_up`

Indicates whether or not the game menu is up.

4.2.4.14 menu_frame

`sf::RectangleShape ContextMenu::menu_frame`

The frame of the context menu.

4.2.4.15 message_hub_ptr

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

A pointer to the message hub.

4.2.4.16 position_x

double ContextMenu::position_x

The position of the object.

4.2.4.17 position_y

double ContextMenu::position_y

The position of the object.

4.2.4.18 render_window_ptr

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

A pointer to the render window.

4.2.4.19 visual_screen

sf::RectangleShape ContextMenu::visual_screen

The context menu screen for visuals.

4.2.4.20 visual_screen_frame_bottom

sf::ConvexShape ContextMenu::visual_screen_frame_bottom

The bottom framing of the visual screen.

4.2.4.21 visual_screen_frame_left

`sf::ConvexShape ContextMenu::visual_screen_frame_left`

The left framing of the visual screen.

4.2.4.22 visual_screen_frame_right

`sf::ConvexShape ContextMenu::visual_screen_frame_right`

The right framing of the visual screen.

4.2.4.23 visual_screen_frame_top

`sf::ConvexShape ContextMenu::visual_screen_frame_top`

The top framing of the visual screen.

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

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

4.3 DieselGenerator Class Reference

A settlement class (child class of TileImprovement).

`#include <DieselGenerator.h>`

Inheritance diagram for DieselGenerator:
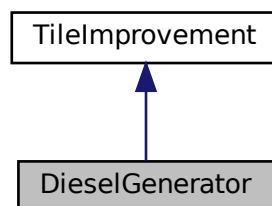
Collaboration diagram for DieselGenerator:
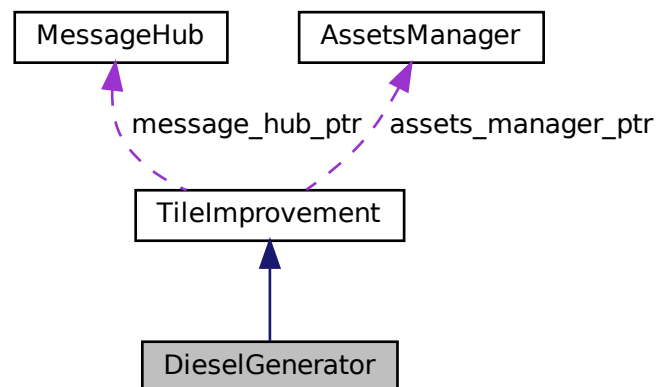
Public Member Functions

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

  Constructor for the DieselGenerator class.
- void processEvent (void)

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

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

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

  Destructor for the DieselGenerator class.

Public Attributes

- bool skip_smoke_processing

  A boolean which indicates whether or not to skip smoke processing.
- double smoke_da

  The per frame delta in smoke particle alpha value.
- double smoke_dx

  The per frame delta in smoke particle x position.
- double smoke_dy

  The per frame delta in smoke particle y position.
- double smoke_prob

  The probability of spawning a new smoke prob in any given frame.
- std::list< sf::Sprite > smoke_sprite_list

  A list of smoke sprite (for chimney animation).

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.3.1 Detailed Description

A settlement class (child class of TileImprovement).

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DieselGenerator()

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

Constructor for the DieselGenerator class.

Ref: Wikipedia [2023]

Parameters

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.

```
212    :
213 TileImprovement(
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
```

```
221 {
222     //  1. set attributes
223
224     //  1.1. private
225     //...
226
227     //  1.2. public
228     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
229
230     this->is_running = false;
231     this->skip_smoke_processing = true;
232
233     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
234     this->smoke_dx = 5 * SECONDS_PER_FRAME;
235     this->smoke_dy = -10 * SECONDS_PER_FRAME;
236     this->smoke_prob = 8 * SECONDS_PER_FRAME;
237
238     this->smoke_sprite_list = {};
239
240     this->tile_improvement_string = "DIESEL GEN";
241
242     this->__setUpTileImprovementSpriteAnimated();
243
244     std::cout « "DieselGenerator constructed at " « this « std::endl;
245
246     return;
247 }   /* DieselGenerator() */
```

4.3.2.2 ∼DieselGenerator()

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

Destructor for the DieselGenerator class.

```
356 {
357     std::cout « "DieselGenerator at " « this « " destroyed" « std::endl;
358
359     return;
360 }   /* ∼DieselGenerator() */
```

4.3.3 Member Function Documentation

4.3.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

```
114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118
119         default: {
120             // do nothing!
121
122             break;
123         }
124     }
125
126     return;
127 }   /* __handleKeyPressEvents() */
```

4.3.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146
147             break;
148         }
149
150
151         case (sf::Mouse::Right): {
152             //...
153
154             break;
155         }
156
157
158         default: {
159             // do nothing!
160
161             break;
162         }
163     }
164
165     return;
166 }   /* __handleMouseButtonEvents() */
```

4.3.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet(
70         *(this->assets_manager_ptr->getTexture("diesel generator"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("diesel generator")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 }   /* __setUpTileImprovementSpriteAnimated() */
```

4.3.3.4   draw()

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

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

Reimplemented from TileImprovement.

```
307 {
308     //  1. if just built, call base method and return
309     if (this->just_built) {
310         TileImprovement :: draw();
311
312         return;
313     }
314
315
316     //  1. draw first element of animated sprite
317     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
318
319
320     //  2. draw second element of animated sprite
321     if (this->is_running) {
322         //...
323     }
324
325     else {
326         //...
327     }
328
329     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
330
331
332     //  3. draw smoke effects
333     if (this->is_running) {
334         //...
335     }
336
337     //...
338
339     this->frame++;
340     return;
341 }   /* draw() */
```

4.3.3.5   processEvent()

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

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

Reimplemented from TileImprovement.

```
262 {
263     if (this->event_ptr->type == sf::Event::KeyPressed) {
264         this->__handleKeyPressEvents();
265     }
266
267     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
268         this->__handleMouseButtonEvents();
269     }
270
271     return;
272 }   /* processEvent() */
```

4.3.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
287 {
288     //...
289
290     return;
291 } /* processMessage() */
```

4.3.4 Member Data Documentation

4.3.4.1 skip_smoke_processing

```
bool DieselGenerator::skip_smoke_processing
```

A boolean which indicates whether or not to skip smoke processing.

4.3.4.2 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.3 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.4 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.5 smoke_prob

```
double DieselGenerator::smoke_prob
```

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

4.3.4.6 smoke_sprite_list

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

A list of smoke sprite (for chimney animation).

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

- header/DieselGenerator.h
- source/DieselGenerator.cpp

4.4 EnergyStorageSystem Class Reference

A settlement class (child class of TileImprovement).

```
#include <EnergyStorageSystem.h>
```

Inheritance diagram for EnergyStorageSystem:

```
TileImprovement
      ▲
      │
EnergyStorageSystem
```
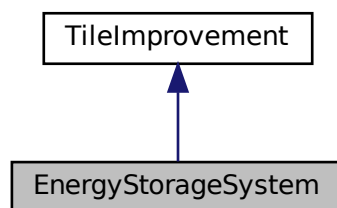
Collaboration diagram for EnergyStorageSystem:
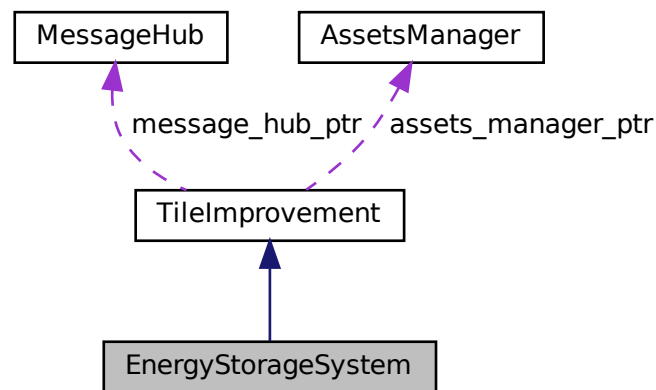
Public Member Functions

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

  Constructor for the EnergyStorageSystem class.
- void processEvent (void)

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

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

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

  Destructor for the EnergyStorageSystem class.

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.4.1 Detailed Description

A settlement class (child class of TileImprovement).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 EnergyStorageSystem()

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

Constructor for the EnergyStorageSystem class.

Ref: Wikipedia [2023]

Parameters

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.

```
201   :
202 TileImprovement(
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
210 {
211     //  1. set attributes
212
213     //  1.1. private
214     //...
215
216     //  1.2. public
217     this->tile_improvement_type = TileImprovementType :: ENERGY_STORAGE_SYSTEM;
218
219     this->is_running = false;
220
221     this->tile_improvement_string = "ENERGY STORAGE";
222
223     this->__setUpTileImprovementSpriteStatic();
224
225     std::cout « "EnergyStorageSystem constructed at " « this « std::endl;
226
227     return;
228 }   /* EnergyStorageSystem() */
```

4.4.2.2 ∼EnergyStorageSystem()

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

Destructor for the EnergyStorageSystem class.

```
317 {
318     std::cout « "EnergyStorageSystem at " « this « " destroyed" « std::endl;
319
320     return;
321 }   /* ~EnergyStorageSystem() */
```

4.4.3 Member Function Documentation

4.4.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

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

4.4.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139
140         case (sf::Mouse::Right): {
141             //...
142
143             break;
144         }
145
146
147         default: {
148             // do nothing!
149
150             break;
151         }
152     }
153
154     return;
155 }   /* __handleMouseButtonEvents() */
```

4.4.3.3 __setUpTileImprovementSpriteStatic()

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

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

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

4.4.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
288 {
289     //  1. if just built, call base method and return
290     if (this->just_built) {
291         TileImprovement :: draw();
292
293         return;
294     }
295
296
297     //  1. draw static sprite
298     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
299
300     this->frame++;
301     return;
302 }   /* draw() */
```

4.4.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
243 {
244     if (this->event_ptr->type == sf::Event::KeyPressed) {
245         this->__handleKeyPressEvents();
246     }
247
248     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
249         this->__handleMouseButtonEvents();
250     }
251
252     return;
253 }   /* processEvent() */
```

4.4.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
268 {
269     //...
270
271     return;
272 }   /* processMessage() */
```

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

- header/EnergyStorageSystem.h
- source/EnergyStorageSystem.cpp

4.5 Game Class Reference

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

```
#include <Game.h>
```

Collaboration diagram for Game:
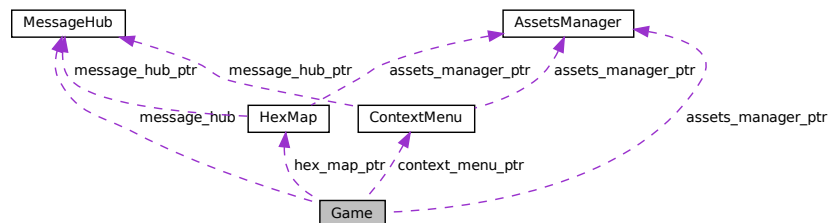
Public Member Functions

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

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

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

    Destructor for the Game class.

Public Attributes

- GamePhase game_phase

    The current phase of the game.
- bool quit_game

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

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

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

    The current frame of the game.
- double time_since_start_s

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

    Current game year.
- int month

    Current game month.
- int population

    Current population.
- int credits

    Current balance of credits.
- int demand_MWh

    Current energy demand [MWh].
- int cumulative_emissions_tonnes

    Cumulative emissions [tonnes] (1 tonne = 1000 kg).
- int turn = 0

    The current game turn.
- sf::Clock clock

    The game clock.
- sf::Event event

    The game events class.
- MessageHub message_hub

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

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

    Pointer to the context menu.

Private Member Functions

- void __toggleFrameClockOverlay (void)

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

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

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

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

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

- void __sendGameStateMessage (void)

  Helper method to format and send a game state message.

- void __insufficientCreditsAlarm (void)

  Helper method to sound and display and insufficient credits alarm.

- void __drawFrameClockOverlay (void)

  Helper method to draw frame clock overlay.

- void __drawHUD (void)

  Helper method to heads-up display (HUD).

- void __draw (void)

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

Private Attributes

- sf::RenderWindow ∗ render_window_ptr

  A pointer to the render window.

- AssetsManager ∗ assets_manager_ptr

  A pointer to the assets manager.

4.5.1 Detailed Description

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

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Game()

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

Constructor for the [Game]() class.

```
702 {
703     //  1. set attributes
704
705     //  1.1. private
706     this->render_window_ptr = render_window_ptr;
707
708     this->assets_manager_ptr = assets_manager_ptr;
709
710     //  1.2. public
711     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
712
713     this->quit_game = false;
714     this->game_loop_broken = false;
715     this->show_frame_clock_overlay = false;
716
717     this->frame = 0;
718     this->time_since_start_s = 0;
719
720     double seconds_since_epoch = time(NULL);
721     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
722
723     this->year = 1970 + (int)years_since_epoch;
```

```
724        this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
725
726        this->population = 0;
727        this->credits = STARTING_CREDITS;
728        this->demand_MWh = 0;
729        this->cumulative_emissions_tonnes = 0;
730
731        this->hex_map_ptr = new HexMap(
732            6,
733            &(this->event),
734            this->render_window_ptr,
735            this->assets_manager_ptr,
736            &(this->message_hub)
737        );
738
739        this->context_menu_ptr = new ContextMenu(
740            &(this->event),
741            this->render_window_ptr,
742            this->assets_manager_ptr,
743            &(this->message_hub)
744        );
745
746        //  2. add message channel(s)
747        this->message_hub.addChannel(GAME_CHANNEL);
748        this->message_hub.addChannel(GAME_STATE_CHANNEL);
749
750        std::cout << "Game constructed at " << this << std::endl;
751
752        return;
753 }   /* Game() */
```

4.5.2.2 ∼Game()

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

Destructor for the Game class.

```
837 {
838        //  1. clean up attributes
839        delete this->hex_map_ptr;
840        delete this->context_menu_ptr;
841
842        std::cout << "Game at " << this << " destroyed" << std::endl;
843
844        return;
845 }   /* ∼Game() */
```

4.5.3 Member Function Documentation

4.5.3.1 __draw()

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

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

```
669 {
670        this->__drawHUD();
671
672        if (this->show_frame_clock_overlay) {
673            this->__drawFrameClockOverlay();
674        }
675
676        return;
677 }   /* draw() */
```

4.5.3.2   __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
495 {
496     std::string frame_clock_string = "FRAME: ";
497     frame_clock_string += std::to_string(this->frame);
498     frame_clock_string += "\nTIME SINCE START [s]: ";
499     frame_clock_string += std::to_string(this->time_since_start_s);
500
501     sf::Text frame_clock_text(
502         frame_clock_string,
503         *(this->assets_manager_ptr->getFont("DroidSansMono")),
504         16
505     );
506
507     sf::RectangleShape frame_clock_backing(
508         sf::Vector2f(
509             1.02 * frame_clock_text.getLocalBounds().width,
510             1.20 * frame_clock_text.getLocalBounds().height
511         )
512     );
513     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
514
515     this->render_window_ptr->draw(frame_clock_backing);
516     this->render_window_ptr->draw(frame_clock_text);
517
518     return;
519 }   /* __drawFrameClockOverlay() */
```

4.5.3.3   __drawHUD()

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

Helper method to heads-up display (HUD).

```
534 {
535     //  1. first line (top)
536     std::string HUD_string = "YEAR: ";
537     HUD_string += std::to_string(this->year);
538
539     HUD_string += "    MONTH: ";
540     HUD_string += std::to_string(this->month);
541
542     HUD_string += "    POPULATION: ";
543     HUD_string += std::to_string(this->population);
544
545     HUD_string += "    CREDITS: ";
546     HUD_string += std::to_string(this->credits);
547     HUD_string += " K";
548
549     HUD_string += "    CURRENT DEMAND: ";
550     HUD_string += std::to_string(this->demand_MWh);
551     HUD_string += " MWh";
552
553     sf::Text HUD_text(
554         HUD_string,
555         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
556         16
557     );
558
559     HUD_text.setPosition(
560         (800 - HUD_text.getLocalBounds().width) / 2,
561         8
562     );
563
564     HUD_text.setFillColor(MONOCHROME_TEXT_GREEN);
565
566     this->render_window_ptr->draw(HUD_text);
567
568
569     //  2. second line (top)
570     HUD_string = "CUMULATIVE EMISSIONS: ";
```

```
571     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
572     HUD_string += " tonnes (CO2e)";
573
574     HUD_string += "    LIFETIME LIMIT: ";
575     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
576     HUD_string += " tonnes (CO2e)";
577
578     HUD_text.setString(HUD_string);
579
580     HUD_text.setPosition(
581         (800 - HUD_text.getLocalBounds().width) / 2,
582         35
583     );
584
585     this->render_window_ptr->draw(HUD_text);
586
587
588     //  3. third line (bottom)
589     HUD_string = "GAME PHASE: ";
590
591     switch (this->game_phase) {
592         case (GamePhase :: BUILD_SETTLEMENT): {
593             HUD_string += "BUILD SETTLEMENT";
594
595             break;
596         }
597
598
599         case (GamePhase :: SYSTEM_MANAGEMENT): {
600             HUD_string += "SYSTEM MANAGEMENT";
601
602             break;
603         }
604
605
606         case (GamePhase :: LOSS_EMISSIONS): {
607             HUD_string += "LOSS (EMISSIONS)";
608
609             break;
610         }
611
612
613         case (GamePhase :: LOSS_DEMAND): {
614             HUD_string += "LOSS (DEMAND)";
615
616             break;
617         }
618
619
620         case (GamePhase :: LOSS_CREDITS): {
621             HUD_string += "LOSS (CREDITS)";
622
623             break;
624         }
625
626
627         case (GamePhase :: VICTORY): {
628             HUD_string += "VICTORY";
629
630             break;
631         }
632
633
634         default: {
635             HUD_string += "???";
636
637             break;
638         }
639     }
640
641     HUD_string += "    TURN: ";
642     HUD_string += std::to_string(this->turn);
643
644     HUD_text.setString(HUD_string);
645
646     HUD_text.setPosition(
647         (800 - HUD_text.getLocalBounds().width) / 2,
648         GAME_HEIGHT - 35
649     );
650
651     this->render_window_ptr->draw(HUD_text);
652
653     return;
654 }   /* __drawHUD() */
```

4.5.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
93 {
94      switch (this->event.key.code) {
95          case (sf::Keyboard::Tilde): {
96              this->__toggleFrameClockOverlay();
97
98              break;
99          }
100
101
102          case (sf::Keyboard::Tab): {
103              this->hex_map_ptr->toggleResourceOverlay();
104
105              break;
106          }
107
108
109          default: {
110              // do nothing!
111
112              break;
113          }
114      }
115
116      return;
117 }   /* __handleKeyPressEvents() */
```

4.5.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
132 {
133      switch (this->event.mouseButton.button) {
134          case (sf::Mouse::Left): {
135              //...
136
137              break;
138          }
139
140
141          case (sf::Mouse::Right): {
142              //...
143
144              break;
145          }
146
147
148          default: {
149              // do nothing!
150
151              break;
152          }
153      }
154
155      return;
156 }   /* __handleMouseButtonEvents() */
```

4.5.3.6 __insufficientCreditsAlarm()

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

Helper method to sound and display and insufficient credits alarm.

```
388 {
389     //  1. sound buzzer
390     this->assets_manager_ptr->getSound("insufficient credits")->play();
391
392     //  2. construct alarm text and backing rectangle
393     sf::Text insufficient_credits_text(
394         "INSUFFICIENT CREDITS",
395         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
396         32
397     );
398
399     insufficient_credits_text.setOrigin(
400         insufficient_credits_text.getLocalBounds().width / 2,
401         insufficient_credits_text.getLocalBounds().height / 2
402     );
403
404     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
405
406     sf::RectangleShape backing_rectangle(
407         sf::Vector2f(
408             1.1 * insufficient_credits_text.getLocalBounds().width,
409             1.5 * insufficient_credits_text.getLocalBounds().height
410         )
411     );
412
413     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
414
415     backing_rectangle.setOrigin(
416         backing_rectangle.getLocalBounds().width / 2,
417         backing_rectangle.getLocalBounds().height / 2
418     );
419
420     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
421
422     //  3. display loop (blocking ~3 seconds)
423     bool red_flag = true;
424     int alarm_frame = 0;
425     double time_since_alarm_s = 0;
426
427     sf::Clock alarm_clock;
428
429     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
430
431
432         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
433
434         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
435             while (this->render_window_ptr->pollEvent(this->event)) {
436                 // do nothing!
437             }
438
439             this->render_window_ptr->clear();
440
441             this->hex_map_ptr->draw();
442             this->context_menu_ptr->draw();
443             this->__draw();
444
445             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
446                 if (red_flag) {
447                     red_flag = false;
448                 }
449
450                 else {
451                     red_flag = true;
452                 }
453             }
454
455             if (red_flag) {
456                 insufficient_credits_text.setFillColor(MONOCHROME_TEXT_RED);
457             }
458
459             else {
460                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
461             }
462
463             this->render_window_ptr->draw(backing_rectangle);
464             this->render_window_ptr->draw(insufficient_credits_text);
465
```

```
466                this->render_window_ptr->display();
467
468                alarm_frame++;
469                this->frame++;
470            }
471
472        // check track status, move to next if stopped
473        if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
474            this->assets_manager_ptr->nextTrack();
475            this->assets_manager_ptr->playTrack();
476        }
477    }
478
479    return;
480 }   /* __insufficientCreditsAlarm( */
```

4.5.3.7 __processEvent()

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

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

```
172 {
173    if (this->event.type == sf::Event::Closed) {
174        this->quit_game = true;
175        this->game_loop_broken = true;
176    }
177
178    if (this->event.type == sf::Event::KeyPressed) {
179        this->__handleKeyPressEvents();
180    }
181
182    if (this->event.type == sf::Event::MouseButtonPressed) {
183        this->__handleMouseButtonEvents();
184    }
185
186    return;
187 }   /* __processEvent() */
```

4.5.3.8 __processMessage()

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

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

```
285 {
286    if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
287        Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
288
289        if (game_channel_message.subject == "quit game") {
290            this->quit_game = true;
291            this->game_loop_broken = true;
292
293            std::cout << "Quit game message received by " << this << std::endl;
294            this->message_hub.popMessage(GAME_CHANNEL);
295        }
296
297        if (game_channel_message.subject == "restart game") {
298            this->game_loop_broken = true;
299
300            std::cout << "Restart game message received by " << this << std::endl;
301            this->message_hub.popMessage(GAME_CHANNEL);
302        }
303
304        if (game_channel_message.subject == "state request") {
305            std::cout << "Game state request message received by " << this << std::endl;
306
307            this->__sendGameStateMessage();
308            this->message_hub.popMessage(GAME_CHANNEL);
```

```
309            }
310
311        if (game_channel_message.subject == "credits spent") {
312            this->credits -= game_channel_message.int_payload["credits spent"];
313
314            std::cout << "Credits spent message (" <<
315                game_channel_message.int_payload["credits spent"] << ") received by "
316                << this << std::endl;
317
318            std::cout << "Current credits (Game): " << this->credits << " K" <<
319                std::endl;
320
321            this->message_hub.popMessage(GAME_CHANNEL);
322        }
323
324        if (game_channel_message.subject == "insufficient credits") {
325            std::cout << "Insufficient credits message received by " << this <<
326                std::endl;
327
328            this->__insufficientCreditsAlarm();
329
330            this->message_hub.popMessage(GAME_CHANNEL);
331        }
332
333        if (game_channel_message.subject == "update game phase") {
334            std::cout << "Update game phase message received by " << this << std::endl;
335
336            if (
337                game_channel_message.string_payload["game phase"] == "system management"
338            ) {
339                this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
340                this->population = STARTING_POPULATION;
341                this->turn++;
342            }
343
344            else if (
345                game_channel_message.string_payload["game phase"] == "loss emissions"
346            ) {
347                this->game_phase = GamePhase :: LOSS_EMISSIONS;
348            }
349
350            else if (
351                game_channel_message.string_payload["game phase"] == "loss demand"
352            ) {
353                this->game_phase = GamePhase :: LOSS_DEMAND;
354            }
355
356            else if (
357                game_channel_message.string_payload["game phase"] == "loss credits"
358            ) {
359                this->game_phase = GamePhase :: LOSS_CREDITS;
360            }
361
362            else if (
363                game_channel_message.string_payload["game phase"] == "victory"
364            ) {
365                this->game_phase = GamePhase :: VICTORY;
366            }
367
368            this->message_hub.popMessage(GAME_CHANNEL);
369        }
370    }
371
372    return;
373 }   /* __processMessage() */
```

4.5.3.9   __sendGameStateMessage()

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

Helper method to format and send a game state message.

```
202 {
203    Message game_state_message;
204
205    game_state_message.channel = GAME_STATE_CHANNEL;
206    game_state_message.subject = "game state";
207
```

```
208        game_state_message.int_payload["year"] = this->year;
209        game_state_message.int_payload["month"] = this->month;
210        game_state_message.int_payload["population"] = this->population;
211        game_state_message.int_payload["credits"] = this->credits;
212        game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
213        game_state_message.int_payload["cumulative_emissions_tonnes"] =
214            this->cumulative_emissions_tonnes;
215
216        switch (this->game_phase) {
217            case (GamePhase :: BUILD_SETTLEMENT): {
218                game_state_message.string_payload["game phase"] = "build settlement";
219
220                break;
221            }
222
223
224            case (GamePhase :: SYSTEM_MANAGEMENT): {
225                game_state_message.string_payload["game phase"] = "system management";
226
227                break;
228            }
229
230
231            case (GamePhase :: LOSS_EMISSIONS): {
232                game_state_message.string_payload["game phase"] = "loss emissions";
233
234                break;
235            }
236
237
238            case (GamePhase :: LOSS_DEMAND): {
239                game_state_message.string_payload["game phase"] = "loss demand";
240
241                break;
242            }
243
244
245            case (GamePhase :: LOSS_CREDITS): {
246                game_state_message.string_payload["game phase"] = "loss credits";
247
248                break;
249            }
250
251
252            case (GamePhase :: VICTORY): {
253                game_state_message.string_payload["game phase"] = "victory";
254
255                break;
256            }
257
258
259            default: {
260                // do nothing!
261
262                break;
263            }
264        }
265
266        this->message_hub.sendMessage(game_state_message);
267
268        std::cout << "Game state message sent by " << this << std::endl;
269        return;
270 }    /* __sendGameStateMessage() */
```

4.5.3.10 __toggleFrameClockOverlay()

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

Helper method to toggle frame clock overlay.

```
68 {
69     if (this->show_frame_clock_overlay) {
70         this->show_frame_clock_overlay = false;
71     }
72
73     else {
74         this->show_frame_clock_overlay = true;
75     }
```

```
76
77      return;
78 }    /* __toggleFrameClockOverlay() */
```

4.5.3.11 run()

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

Method to run game (defines game loop).

Returns

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

```
771 {
772      //  1. play brand animation
773      //...
774
775      //  2. show splash screen
776      //...
777
778      //  3. start game loop
779      while (not this->game_loop_broken) {
780          this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
781
782          if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
783              //  6.1. process events
784              while (this->render_window_ptr->pollEvent(this->event)) {
785                  this->hex_map_ptr->processEvent();
786                  this->context_menu_ptr->processEvent();
787                  this->__processEvent();
788              }
789
790
791              //  6.2. process messages
792              while (this->message_hub.hasTraffic()) {
793                  this->hex_map_ptr->processMessage();
794                  this->context_menu_ptr->processMessage();
795                  this->__processMessage();
796              }
797
798
799              //  6.3. draw frame
800              this->render_window_ptr->clear();
801
802              this->hex_map_ptr->draw();
803              this->context_menu_ptr->draw();
804              this->__draw();
805
806              this->render_window_ptr->display();
807
808
809              //  6.4. increment frame
810              this->frame++;
811          }
812
813          // check track status, move to next if stopped
814          if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
815              this->assets_manager_ptr->nextTrack();
816              this->assets_manager_ptr->playTrack();
817          }
818
819      }
820
821      return this->quit_game;
822 }    /* run() */
```

4.5.4 Member Data Documentation

4.5.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.5.4.2 clock

`sf::Clock Game::clock`

The game clock.

4.5.4.3 context_menu_ptr

[ContextMenu](#)* Game::context_menu_ptr

Pointer to the context menu.

4.5.4.4 credits

`int Game::credits`

Current balance of credits.

4.5.4.5 cumulative_emissions_tonnes

`int Game::cumulative_emissions_tonnes`

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

4.5.4.6 demand_MWh

`int Game::demand_MWh`

Current energy demand [MWh].

4.5.4.7 event

`sf::Event Game::event`

The game events class.

4.5.4.8 frame

`unsigned long long int Game::frame`

The current frame of the game.

4.5.4.9 game_loop_broken

`bool Game::game_loop_broken`

Boolean indicating whether or not the game loop is broken.

4.5.4.10 game_phase

`GamePhase Game::game_phase`

The current phase of the game.

4.5.4.11 hex_map_ptr

`HexMap* Game::hex_map_ptr`

Pointer to the hex map (defines game world).

4.5.4.12 message_hub

`MessageHub Game::message_hub`

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

4.5.4.13 month

`int Game::month`

Current game month.

4.5.4.14 population

`int Game::population`

Current population.

4.5.4.15 quit_game

`bool Game::quit_game`

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

4.5.4.16 render_window_ptr

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

A pointer to the render window.

4.5.4.17 show_frame_clock_overlay

`bool Game::show_frame_clock_overlay`

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

4.5.4.18 time_since_start_s

`double Game::time_since_start_s`

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

4.5.4.19 turn

`int Game::turn = 0`

The current game turn.

4.5.4.20 year

`int Game::year`

Current game year.

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

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

4.6 HexMap Class Reference

A class which defines a hex map of hex tiles.

`#include <HexMap.h>`

Collaboration diagram for HexMap:
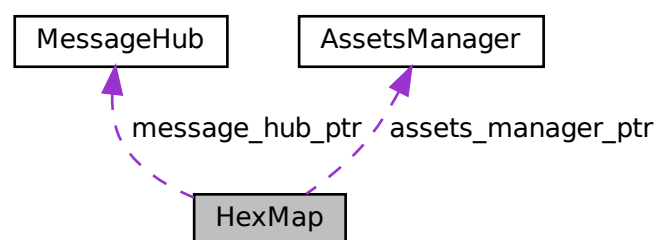
Public Member Functions

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

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

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

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

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

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

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

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

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

    Destructor for the HexMap class.

Public Attributes

- bool show_resource

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

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

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

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

    The current frame of this object.
- double position_x

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

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

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

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

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

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

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

    A vector of hex tiles, in drawing order.

Private Member Functions

- void __setUpGlassScreen (void)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    Helper method to format and send message on no tile selected.
- void __assessNeighbours (HexTile ∗)

    Helper method to assess all neighbours of the given tile.

Private Attributes

- sf::Event ∗ event_ptr

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

    A pointer to the render window.
- AssetsManager ∗ assets_manager_ptr

    A pointer to the assets manager.
- MessageHub ∗ message_hub_ptr

    A pointer to the message hub.

4.6.1 Detailed Description

A class which defines a hex map of hex tiles.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 HexMap()

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

Constructor (intended) for the HexMap class.

Parameters

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.

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

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

4.6.2.2 ∼HexMap()

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

Destructor for the HexMap class.

```
1447 {
1448     this->clear();
1449
1450     std::cout « "HexMap at " « this « " destroyed" « std::endl;
1451
1452     return;
1453 }    /* ~HexMap() */
```

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

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

Helper method to assemble the hex map.

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

4.6.3.2 __assessNeighbours()

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

Helper method to assess all neighbours of the given tile.

Parameters

Pointer	to the tile whose neighbours are to be assessed.

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

4.6.3.3 __buildDrawOrderVector()

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

Helper method to build tile drawing order vector.

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

4.6.3.4 __enforceOceanContinuity()

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

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

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

4.6.3.5 __getMajorityTileType()

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

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

Parameters

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.

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

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

4.6.3.6 __getNeighboursVector()

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

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

Parameters

hex_ptr	A pointer to the given tile.

Returns

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

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

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

4.6.3.7 __getNoise()

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

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

Parameters

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].

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

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

4.6.3.8 __getSelectedTile()

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

Helper method to get pointer to selected tile.

Returns

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

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

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

4.6.3.9 __getValidMapIndexPositions()

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

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

Parameters

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.

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

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

4.6.3.10 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

4.6.3.11 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

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

4.6.3.12 __isLakeTouchingOcean()

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

4.6.3.13 __layTiles()

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

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

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

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

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

4.6.3.14 __procedurallyGenerateTileResources()

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

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

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

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

4.6.3.15 __procedurallyGenerateTileTypes()

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

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

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

4.6.3.16   __sendNoTileSelectedMessage()

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

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

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

4.6.3.17   __setUpGlassScreen()

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

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

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

4.6.3.18   __smoothTileTypes()

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

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

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

4.6.3.19 assess()

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

Method to assess the resource of the selected tile.

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

4.6.3.20 clear()

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

Method to clear the hex map.

```
1409 {
1410     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1411     std::map<double, HexTile*>::iterator hex_map_iter_y;
1412     for (
1413         hex_map_iter_x = this->hex_map.begin();
1414         hex_map_iter_x != this->hex_map.end();
1415         hex_map_iter_x++
1416     ) {
1417         for (
1418             hex_map_iter_y = hex_map_iter_x->second.begin();
1419             hex_map_iter_y != hex_map_iter_x->second.end();
1420             hex_map_iter_y++
1421         ) {
1422             delete hex_map_iter_y->second;
1423         }
1424     }
1425     this->hex_map.clear();
1426
1427     this->tile_position_x_vec.clear();
1428     this->tile_position_y_vec.clear();
1429     this->border_tiles_vec.clear();
1430
1431     return;
1432 }   /* clear() */
```

4.6.3.21 draw()

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

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

```
1348 {
1349     //  1. draw background
1350     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1351     glass_screen_colour.a = 255;
1352     this->glass_screen.setFillColor(glass_screen_colour);
1353
1354     this->render_window_ptr->draw(this->glass_screen);
1355
1356     //  2. draw tiles in drawing order
1357     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1358         this->hex_draw_order_vec[i]->draw();
1359     }
1360
1361     //  3. redraw selected tile
```

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

4.6.3.22   processEvent()

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

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

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

4.6.3.23 processMessage()

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

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

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

4.6.3.24 reroll()

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

Method to re-roll the hex map.

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

4.6.3.25 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

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

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

4.6.4 Member Data Documentation

4.6.4.1 assets_manager_ptr

AssetsManager* HexMap::assets_manager_ptr [private]

A pointer to the assets manager.

4.6.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

4.6.4.3 event_ptr

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

A pointer to the event class.

4.6.4.4 frame

unsigned long long int HexMap::frame

The current frame of this object.

4.6.4.5 glass_screen

`sf::RectangleShape HexMap::glass_screen`

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

4.6.4.6 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

4.6.4.7 hex_map

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

A position-indexed, nested map of hex tiles.

4.6.4.8 message_hub_ptr

`MessageHub* HexMap::message_hub_ptr  [private]`

A pointer to the message hub.

4.6.4.9 n_layers

`int HexMap::n_layers`

The number of layers in the hex map.

4.6.4.10 n_tiles

`int HexMap::n_tiles`

The number of tiles in the hex map.

4.6.4.11 position_x

`double HexMap::position_x`

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

4.6.4.12 position_y

`double HexMap::position_y`

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

4.6.4.13 render_window_ptr

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

A pointer to the render window.

4.6.4.14 show_resource

`bool HexMap::show_resource`

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

4.6.4.15 tile_position_x_vec

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

A vector of tile x positions.

4.6.4.16 tile_position_y_vec

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

A vector of tile y position.

4.6.4.17 tile_selected

```
bool HexMap::tile_selected
```

A boolean which indicates if a tile is currently selected.

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

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

4.7 HexTile Class Reference

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

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

Collaboration diagram for HexTile:
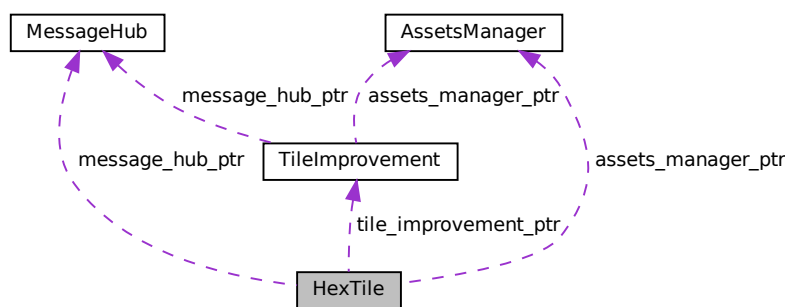
Public Member Functions

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

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

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

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

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

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

    Method to decorate tile.
- void toggleResourceOverlay (void)

    Method to toggle the tile resource overlay.

- void assess (void)

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

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

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

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

    Destructor for the HexTile class.

Public Attributes

- TileType tile_type
- TileResource tile_resource
- bool show_node

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

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

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

    A boolean which triggers a resource assessment notification.
- bool is_selected

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

    A boolean which indicates whether or not to draw a tile explosion.
- bool decoration_cleared

    A boolean which indicates if the tile decoration has been cleared.
- bool has_improvement

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

    A pointer to the improvement for this tile.
- bool build_menu_open

    A boolean which indicates if the tile build menu is open.
- size_t explosion_frame

    The current frame of the explosion animation.
- unsigned long long int frame

    The current frame of this object.
- int credits

    The current balance of credits.
- double position_x

    The x position of the tile.
- double position_y

    The y position of the tile.
- double major_radius

    The radius of the smallest bounding circle.
- double minor_radius

    The radius of the largest inscribed circle.
- std::string game_phase

The current phase of the game.
- sf::CircleShape node_sprite

    A circle shape to mark the tile node.
- sf::ConvexShape tile_sprite

    A convex shape which represents the tile.
- sf::ConvexShape select_outline_sprite

    A convex shape which outlines the tile when selected.
- sf::CircleShape resource_chip_sprite

    A circle shape which represents a resource chip.
- sf::Text resource_text

    A text representation of the resource.
- sf::Sprite tile_decoration_sprite

    A tile decoration sprite.
- sf::Sprite magnifying_glass_sprite

    A magnifying glass sprite.
- std::vector< sf::Sprite > explosion_sprite_reel

    A reel of sprites for a tile explosion animation.
- sf::RectangleShape build_menu_backing

    A backing for the tile build menu.
- sf::Text build_menu_backing_text

    A text label for the build menu.
- std::vector< std::vector< sf::Sprite > > build_menu_options_vec

    A vector of sprites for illustrating the tile build options.
- std::vector< sf::Text > build_menu_options_text_vec

    A vector of text for the tile build options.

Private Member Functions

- void __setUpNodeSprite (void)

    Helper method to set up node sprite.
- void __setUpTileSprite (void)

    Helper method to set up tile sprite.
- void __setUpSelectOutlineSprite (void)

    Helper method to set up select outline sprite.
- void __setUpResourceChipSprite (void)

    Helper method to set up resource chip sprite.
- void __setResourceText (void)

    Helper method to set up resource text.
- void __setUpMagnifyingGlassSprite (void)

    Helper method to set up and position magnifying glass sprite.
- void __setUpTileExplosionReel (void)

    Helper method to set up tile explosion sprite reel.
- void __setUpBuildOption (std::string, std::string)

    Helper method to set up and postion the sprite and text for a build option.
- void __setUpDieselGeneratorBuildOption (void)

    Helper method to set up and position the diesel generator build option.
- void __setUpWindTurbineBuildOption (bool=false, bool=false)

    Helper method to set up and position the wind turbine build option.
- void __setUpSolarPVBuildOption (bool=false)

    Helper method to set up and position the solar PV array build option.

- void __setUpTidalTurbineBuildOption (void)

  Helper method to set up and position the tidal turbine build option.
- void __setUpWaveEnergyConverterBuildOption (void)

  Helper method to set up and position the wave energy converter build option.
- void __setUpEnergyStorageSystemBuildOption (void)

  Helper method to set up and position the wave energy converter build option.
- void __setUpBuildMenu (void)

  Helper method to set up and place build menu assets (drawable).
- void __setIsSelected (bool)

  Helper method to set the is selected attribute (of tile and improvement).
- void __clearDecoration (void)

  Helper method to clear tile decoration.
- bool __isClicked (void)

  Helper method to determine if tile was clicked on.
- void __handleKeyPressEvents (void)

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

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

  Helper method to open the tile improvement build menu.
- void __closeBuildMenu (void)

  Helper method to close the tile improvement build menu.
- void __buildSettlement (void)

  Helper method to build a settlement on this tile.
- void __buildDieselGenerator (void)

  Helper method to build a diesel generator on this tile.
- void __buildSolarPV (void)

  Helper method to build a solar PV array on this tile.
- void __buildWindTurbine (void)

  Helper method to build a wind turbine on this tile.
- void __buildTidalTurbine (void)

  Helper method to build a tidal turbine on this tile.
- void __buildWaveEnergyConverter (void)

  Helper method to build a wave energy converter on this tile.
- void __buildEnergyStorage (void)

  Helper method to build an energy storage system on this tile.
- void __sendTileSelectedMessage (void)

  Helper method to format and send message on tile selection.
- std::string __getTileCoordsSubstring (void)

  Helper method to assemble and return tile coordinates substring.
- std::string __getTileTypeSubstring (void)

  Helper method to assemble and return tile type substring.
- std::string __getTileResourceSubstring (void)

  Helper method to assemble and return tile resource substring.
- std::string __getTileImprovementSubstring (void)

  Helper method to assemble and return the tile improvement substring.
- std::string __getTileOptionsSubstring (void)

  Helper method to assemble and return tile options substring.
- void __sendTileStateMessage (void)

  Helper method to format and send tile state message.
- void __sendAssessNeighboursMessage (void)

Helper method to format and send assess neighbours message.
- void __sendGameStateRequest (void)

    Helper method to format and send a game state request (message).
- void __sendUpdateGamePhaseMessage (std::string)

    Helper method to format and send update game phase message.
- void __sendCreditsSpentMessage (int)

    Helper method to format and send a credits spent message.
- void __sendInsufficientCreditsMessage (void)

    Helper method to format and send an insufficient credits message.

Private Attributes

- sf::Event ∗ event_ptr

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

    A pointer to the render window.
- AssetsManager ∗ assets_manager_ptr

    A pointer to the assets manager.
- MessageHub ∗ message_hub_ptr

    A pointer to the message hub.

4.7.1 Detailed Description

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

4.7.2 Constructor & Destructor Documentation

4.7.2.1 HexTile()

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

Constructor for the HexTile class.

Ref: Wikipedia [2023]

Parameters

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.

```
2181 {
2182     // 1. set attributes
2183
2184     // 1.1. private
2185     this->event_ptr = event_ptr;
2186     this->render_window_ptr = render_window_ptr;
2187
2188     this->assets_manager_ptr = assets_manager_ptr;
2189     this->message_hub_ptr = message_hub_ptr;
2190
2191     // 1.2. public
2192     this->show_node = false;
2193     this->show_resource = false;
2194     this->resource_assessed = false;
2195     this->resource_assessment = false;
2196     this->is_selected = false;
2197     this->draw_explosion = false;
2198
2199     this->decoration_cleared = false;
2200     this->has_improvement = false;
2201     this->tile_improvement_ptr = NULL;
2202
2203     this->build_menu_open = false;
2204
2205     this->explosion_frame = 0;
2206
2207     this->frame = 0;
2208     this->credits = 0;
2209
2210     this->position_x = position_x;
2211     this->position_y = position_y;
2212
2213     this->major_radius = 32;
2214     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2215
2216     this->game_phase = "build settlement";
2217
2218     // 2. set up and position drawable attributes
2219     this->__setUpNodeSprite();
2220     this->__setUpTileSprite();
2221     this->__setUpSelectOutlineSprite();
2222     this->__setUpResourceChipSprite();
2223     this->__setResourceText();
2224     this->__setUpMagnifyingGlassSprite();
2225     this->__setUpTileExplosionReel();
2226
2227     // 3. set tile type and resource (default to none type and average)
2228     this->setTileType(TileType :: NONE_TYPE);
2229     this->setTileResource(TileResource :: AVERAGE);
2230
2231     std::cout « "HexTile constructed at " « this « std::endl;
2232
2233     return;
2234 }   /* HexTile() */
```

4.7.2.2 ∼HexTile()

```
HexTile::~HexTile (
            void  )
```

Destructor for the HexTile class.

```
2765 {
2766     if (this->tile_improvement_ptr != NULL) {
2767         delete this->tile_improvement_ptr;
2768     }
2769
2770     std::cout « "HexTile at " « this « " destroyed" « std::endl;
2771
2772     return;
2773 }   /* ~HexTile() */
```

4.7.3 Member Function Documentation

4.7.3.1 __buildDieselGenerator()

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

Helper method to build a diesel generator on this tile.

```
1361 {
1362     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1363
1364     if (this->credits < build_cost) {
1365         std::cout « "Cannot build diesel generator: insufficient credits (need "
1366             « build_cost « " K)" « std::endl;
1367
1368         this->__sendInsufficientCreditsMessage();
1369         return;
1370     }
1371
1372     this->tile_improvement_ptr = new DieselGenerator(
1373         this->position_x,
1374         this->position_y,
1375         this->event_ptr,
1376         this->render_window_ptr,
1377         this->assets_manager_ptr,
1378         this->message_hub_ptr
1379     );
1380
1381     this->has_improvement = true;
1382     this->__closeBuildMenu();
1383
1384     this->__sendCreditsSpentMessage(build_cost);
1385     this->__sendTileStateMessage();
1386     this->__sendGameStateRequest();
1387
1388     return;
1389 }   /* __buildDieselGenerator() */
```

4.7.3.2 __buildEnergyStorage()

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

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

```
1604 {
1605     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1606
1607     if (this->credits < build_cost) {
1608         std::cout « "Cannot build energy storage system: insufficient credits (need "
1609             « build_cost « " K)" « std::endl;
1610
1611         this->__sendInsufficientCreditsMessage();
1612         return;
1613     }
1614
1615     this->tile_improvement_ptr = new EnergyStorageSystem(
1616         this->position_x,
1617         this->position_y,
1618         this->event_ptr,
1619         this->render_window_ptr,
1620         this->assets_manager_ptr,
1621         this->message_hub_ptr
1622     );
1623
1624     this->has_improvement = true;
1625     this->__closeBuildMenu();
1626
1627     this->__sendCreditsSpentMessage(build_cost);
1628     this->__sendTileStateMessage();
1629     this->__sendGameStateRequest();
1630
1631     return;
1632 }   /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

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

Helper method to build a settlement on this tile.

```
1315 {
1316     if (this->credits < BUILD_SETTLEMENT_COST) {
1317         std::cout << "Cannot build settlement: insufficient credits (need "
1318             << BUILD_SETTLEMENT_COST << " K)" << std::endl;
1319
1320         this->__sendInsufficientCreditsMessage();
1321         return;
1322     }
1323
1324     this->__clearDecoration();
1325
1326     this->tile_improvement_ptr = new Settlement(
1327         this->position_x,
1328         this->position_y,
1329         this->event_ptr,
1330         this->render_window_ptr,
1331         this->assets_manager_ptr,
1332         this->message_hub_ptr
1333     );
1334
1335     this->has_improvement = true;
1336
1337     this->assess();
1338     this->__sendAssessNeighboursMessage();
1339
1340     this->__sendUpdateGamePhaseMessage("system management");
1341     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1342     this->__sendTileStateMessage();
1343     this->__sendGameStateRequest();
1344
1345     return;
1346 }   /* __buildSettlement() */
```

4.7.3.4 __buildSolarPV()

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

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

```
1404 {
1405     int build_cost = SOLAR_PV_BUILD_COST;
1406
1407     if (this->tile_type == TileType :: LAKE) {
1408         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1409     }
1410
1411     if (this->credits < build_cost) {
1412         std::cout << "Cannot build solar PV array: insufficient credits (need "
1413             << build_cost << " K)" << std::endl;
1414
1415         this->__sendInsufficientCreditsMessage();
1416         return;
1417     }
1418
1419     this->tile_improvement_ptr = new SolarPV(
1420         this->position_x,
1421         this->position_y,
1422         this->event_ptr,
1423         this->render_window_ptr,
1424         this->assets_manager_ptr,
1425         this->message_hub_ptr
1426     );
1427
1428     this->has_improvement = true;
1429     this->__closeBuildMenu();
1430
1431     if (this->tile_type == TileType :: LAKE) {
1432         this->decoration_cleared = true;
1433         this->assets_manager_ptr->getSound("splash")->play();
```

```
1434     }
1435
1436     this->__sendCreditsSpentMessage(build_cost);
1437     this->__sendTileStateMessage();
1438     this->__sendGameStateRequest();
1439
1440     return;
1441 }   /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

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

Helper method to build a tidal turbine on this tile.

```
1514 {
1515     int build_cost = TIDAL_TURBINE_BUILD_COST;
1516
1517     if (this->credits < build_cost) {
1518         std::cout « "Cannot build tidal turbine: insufficient credits (need "
1519             « build_cost « " K)" « std::endl;
1520
1521         this->__sendInsufficientCreditsMessage();
1522         return;
1523     }
1524
1525     this->tile_improvement_ptr = new TidalTurbine(
1526         this->position_x,
1527         this->position_y,
1528         this->event_ptr,
1529         this->render_window_ptr,
1530         this->assets_manager_ptr,
1531         this->message_hub_ptr
1532     );
1533
1534     this->has_improvement = true;
1535     this->decoration_cleared = true;
1536     this->assets_manager_ptr->getSound("splash")->play();
1537     this->__closeBuildMenu();
1538
1539     this->__sendCreditsSpentMessage(build_cost);
1540     this->__sendTileStateMessage();
1541     this->__sendGameStateRequest();
1542
1543     return;
1544 }   /* __buildTidalTurbine() */
```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1559 {
1560     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1561
1562     if (this->credits < build_cost) {
1563         std::cout « "Cannot build wave energy converter: insufficient credits (need "
1564             « build_cost « " K)" « std::endl;
1565
1566         this->__sendInsufficientCreditsMessage();
1567         return;
1568     }
1569
1570     this->tile_improvement_ptr = new WaveEnergyConverter(
1571         this->position_x,
1572         this->position_y,
1573         this->event_ptr,
1574         this->render_window_ptr,
```

```
1575        this->assets_manager_ptr,
1576        this->message_hub_ptr
1577    );
1578
1579    this->has_improvement = true;
1580    this->decoration_cleared = true;
1581    this->assets_manager_ptr->getSound("splash")->play();
1582    this->__closeBuildMenu();
1583
1584    this->__sendCreditsSpentMessage(build_cost);
1585    this->__sendTileStateMessage();
1586    this->__sendGameStateRequest();
1587
1588    return;
1589 }    /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1456 {
1457    int build_cost = WIND_TURBINE_BUILD_COST;
1458
1459    if (
1460        (this->tile_type == TileType :: LAKE) or
1461        (this->tile_type == TileType :: OCEAN)
1462    ) {
1463        build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1464    }
1465
1466    if (this->credits < build_cost) {
1467        std::cout << "Cannot build wind turbine: insufficient credits (need "
1468            << build_cost << " K)" << std::endl;
1469
1470        this->__sendInsufficientCreditsMessage();
1471        return;
1472    }
1473
1474    this->tile_improvement_ptr = new WindTurbine(
1475        this->position_x,
1476        this->position_y,
1477        this->event_ptr,
1478        this->render_window_ptr,
1479        this->assets_manager_ptr,
1480        this->message_hub_ptr
1481    );
1482
1483    this->has_improvement = true;
1484    this->__closeBuildMenu();
1485
1486    if (
1487        (this->tile_type == TileType :: LAKE) or
1488        (this->tile_type == TileType :: OCEAN)
1489    ) {
1490        this->decoration_cleared = true;
1491        this->assets_manager_ptr->getSound("splash")->play();
1492    }
1493
1494    this->__sendCreditsSpentMessage(build_cost);
1495    this->__sendTileStateMessage();
1496    this->__sendGameStateRequest();
1497
1498    return;
1499 }    /* __buildWindTurbine() */
```

4.7.3.8 __clearDecoration()

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

Helper method to clear tile decoration.

```
807 {
808     this->decoration_cleared = true;
809     this->draw_explosion = true;
810
811     switch (this->tile_type) {
812         case (TileType :: FOREST): {
813             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
814
815             break;
816         }
817
818
819         case (TileType :: MOUNTAINS): {
820             this->assets_manager_ptr->getSound("clear mountains tile")->play();
821
822             break;
823         }
824
825
826         case (TileType :: PLAINS): {
827             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
828
829             break;
830         }
831
832
833         default: {
834             // do nothing!
835
836             break;
837         }
838     }
839
840     return;
841 }   /* __clearDecoration() */
```

4.7.3.9 __closeBuildMenu()

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

Helper method to close the tile improvement build menu.

```
1290 {
1291     if (not this->build_menu_open) {
1292         return;
1293     }
1294
1295     this->build_menu_open = false;
1296     this->assets_manager_ptr->getSound("build menu close")->play();
1297
1298     return;
1299 }   /* __closeBuildMenu() */
```

4.7.3.10 __getTileCoordsSubstring()

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

Helper method to assemble and return tile coordinates substring.

Returns

Tile coordinates substring.

```
1673 {
1674     std::string coords_substring = "TILE COORDS:  (";
1675     coords_substring += std::to_string(int(this->position_x - 400));
1676     coords_substring += ", ";
1677     coords_substring += std::to_string(int(this->position_y - 400));
1678     coords_substring += ")\n";
1679
1680     return coords_substring;
1681 }   /* __getTileCoordsSubstring() */
```

4.7.3.11 __getTileImprovementSubstring()

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

Helper method to assemble and return the tile improvement substring.

Returns

Tile improvement substring.

```
1832 {
1833     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1834
1835     if (this->has_improvement) {
1836         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1837         improvement_substring += "\n";
1838     }
1839
1840     else {
1841         improvement_substring += "NONE\n";
1842     }
1843
1844     return improvement_substring;
1845 }   /* __getTileImprovementSubstring() */
```

4.7.3.12 __getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

```
1862 {
1863     //                    32 char x 17 line console "--------------------------------\n";
1864     std::string options_substring              = "    **** TILE OPTIONS ****    \n";
1865     options_substring                         += "                              \n";
1866
1867     if (this->game_phase == "build settlement") {
1868         if (
1869             (this->tile_type != TileType :: OCEAN) and
1870             (this->tile_type != TileType :: LAKE)
1871         ) {
1872             options_substring += "[B]:  BUILD SETTLEMENT (";
1873             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
1874             options_substring += " K)";
1875         }
```

```
1876        }
1877
1878
1879        else if (this->game_phase == "system management") {
1880            if (this->has_improvement) {
1881                /*
1882                options_substring.clear();
1883                options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
1884                */
1885            }
1886
1887
1888            else if (not this->resource_assessed) {
1889                options_substring += "[A]:  ASSESS RESOURCE (";
1890                options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
1891                options_substring += " K)\n";
1892            }
1893
1894
1895            else if (
1896                (not this->decoration_cleared) and
1897                (this->tile_type != TileType :: OCEAN) and
1898                (this->tile_type != TileType :: LAKE)
1899            ) {
1900                options_substring += "[C]:  CLEAR TILE (";
1901
1902                switch (this->tile_type) {
1903                    case (TileType :: FOREST): {
1904                        options_substring += std::to_string(CLEAR_FOREST_COST);
1905
1906                        break;
1907                    }
1908
1909
1910                    case (TileType :: MOUNTAINS): {
1911                        options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
1912
1913                        break;
1914                    }
1915
1916
1917                    case (TileType :: PLAINS): {
1918                        options_substring += std::to_string(CLEAR_PLAINS_COST);
1919
1920                        break;
1921                    }
1922
1923
1924                    default: {
1925                        //do nothing!
1926
1927                        break;
1928                    }
1929                }
1930
1931                options_substring += " K)\n";
1932            }
1933
1934
1935            else if (
1936                (this->decoration_cleared) or
1937                (this->tile_type == TileType :: OCEAN) or
1938                (this->tile_type == TileType :: LAKE)
1939            ) {
1940                options_substring += "[B]: OPEN BUILD MENU\n";
1941            }
1942        }
1943
1944
1945        else if (this->game_phase == "victory") {
1946            options_substring                        += "      **** VICTORY ****        \n";
1947        }
1948
1949
1950        else {
1951            options_substring                        += "      **** LOSS ****        \n";
1952        }
1953
1954        return options_substring;
1955 }  /* __getTileOptionsString() */
```

4.7.3.13 __getTileResourceSubstring()

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

Helper method to assemble and return tile resource substring.

Returns

Tile resource substring.

```
1762 {
1763     std::string resource_substring = "TILE RESOURCE:     ";
1764
1765     if (this->resource_assessed) {
1766         switch (this->tile_resource) {
1767             case (TileResource :: POOR): {
1768                 resource_substring += "POOR\n";
1769
1770                 break;
1771             }
1772
1773
1774             case (TileResource ::BELOW_AVERAGE): {
1775                 resource_substring += "BELOW AVERAGE\n";
1776
1777                 break;
1778             }
1779
1780
1781             case (TileResource :: AVERAGE): {
1782                 resource_substring += "AVERAGE\n";
1783
1784                 break;
1785             }
1786
1787
1788             case (TileResource :: ABOVE_AVERAGE): {
1789                 resource_substring += "ABOVE AVERAGE\n";
1790
1791                 break;
1792             }
1793
1794
1795             case (TileResource :: GOOD): {
1796                 resource_substring += "GOOD\n";
1797
1798                 break;
1799             }
1800
1801
1802             default: {
1803                 resource_substring += "???\n";
1804
1805                 break;
1806             }
1807         }
1808     }
1809
1810     else {
1811         resource_substring += "???\n";
1812     }
1813
1814     return resource_substring;
1815 }  /* __getTileResourceSubstring() */
```

4.7.3.14 __getTileTypeSubstring()

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

Helper method to assemble and return tile type substring.

Returns

Tile type substring.

```
1698 {
1699     std::string type_substring = "TILE TYPE:          ";
1700
1701     switch (this->tile_type) {
1702         case (TileType :: FOREST): {
1703             type_substring += "FOREST\n";
1704
1705             break;
1706         }
1707
1708
1709         case (TileType :: LAKE): {
1710             type_substring += "LAKE\n";
1711
1712             break;
1713         }
1714
1715
1716         case (TileType :: MOUNTAINS): {
1717             type_substring += "MOUNTAINS\n";
1718
1719             break;
1720         }
1721
1722
1723         case (TileType :: OCEAN): {
1724             type_substring += "OCEAN\n";
1725
1726             break;
1727         }
1728
1729
1730         case (TileType :: PLAINS): {
1731             type_substring += "PLAINS\n";
1732
1733             break;
1734         }
1735
1736
1737         default: {
1738             type_substring += "???\n";
1739
1740             break;
1741         }
1742     }
1743
1744     return type_substring;
1745 }   /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
890 {
891     if (not this->is_selected) {
892         return;
893     }
894
895
896     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
897         this->__setIsSelected(false);
898     }
899
900
901     if (this->build_menu_open) {
902         switch (this->tile_type) {
903             case (TileType :: FOREST): {
904                 switch (this->event_ptr->key.code) {
905                     case (sf::Keyboard::D): {
906                         this->__buildDieselGenerator();
907
908                         break;
```

```
909                     }
910
911
912                     case (sf::Keyboard::S): {
913                         this->__buildSolarPV();
914
915                         break;
916                     }
917
918
919                     case (sf::Keyboard::W): {
920                         this->__buildWindTurbine();
921
922                         break;
923                     }
924
925
926                     case (sf::Keyboard::E): {
927                         this->__buildEnergyStorage();
928
929                         break;
930                     }
931
932
933                     default: {
934                         // do nothing!
935
936                         break;
937                     }
938                 }
939
940                 break;
941             }
942
943
944             case (TileType :: LAKE): {
945                 switch (this->event_ptr->key.code) {
946                     case (sf::Keyboard::S): {
947                         this->__buildSolarPV();
948
949                         break;
950                     }
951
952
953                     case (sf::Keyboard::W): {
954                         this->__buildWindTurbine();
955
956                         break;
957                     }
958
959
960                     default: {
961                         // do nothing!
962
963                         break;
964                     }
965                 }
966
967                 break;
968             }
969
970
971             case (TileType :: MOUNTAINS): {
972                 switch (this->event_ptr->key.code) {
973                     case (sf::Keyboard::D): {
974                         this->__buildDieselGenerator();
975
976                         break;
977                     }
978
979
980                     case (sf::Keyboard::S): {
981                         this->__buildSolarPV();
982
983                         break;
984                     }
985
986
987                     case (sf::Keyboard::W): {
988                         this->__buildWindTurbine();
989
990                         break;
991                     }
992
993
994                     case (sf::Keyboard::E): {
995                         this->__buildEnergyStorage();
```

```
996
997                          break;
998                      }
999
1000
1001                 default: {
1002                     // do nothing!
1003
1004                     break;
1005                 }
1006             }
1007
1008         break;
1009     }
1010
1011
1012     case (TileType :: OCEAN): {
1013         switch (this->event_ptr->key.code) {
1014             case (sf::Keyboard::W): {
1015                 this->__buildWindTurbine();
1016
1017                 break;
1018             }
1019
1020
1021             case (sf::Keyboard::T): {
1022                 this->__buildTidalTurbine();
1023
1024                 break;
1025             }
1026
1027
1028             case (sf::Keyboard::A): {
1029                 this->__buildWaveEnergyConverter();
1030
1031                 break;
1032             }
1033
1034
1035             default: {
1036                 // do nothing!
1037
1038                 break;
1039             }
1040         }
1041
1042         break;
1043     }
1044
1045
1046     case (TileType :: PLAINS): {
1047         switch (this->event_ptr->key.code) {
1048             case (sf::Keyboard::D): {
1049                 this->__buildDieselGenerator();
1050
1051                 break;
1052             }
1053
1054
1055             case (sf::Keyboard::S): {
1056                 this->__buildSolarPV();
1057
1058                 break;
1059             }
1060
1061
1062             case (sf::Keyboard::W): {
1063                 this->__buildWindTurbine();
1064
1065                 break;
1066             }
1067
1068
1069             case (sf::Keyboard::E): {
1070                 this->__buildEnergyStorage();
1071
1072                 break;
1073             }
1074
1075
1076             default: {
1077                 // do nothing!
1078
1079                 break;
1080             }
1081         }
1082
```

```
1083                    break;
1084                }
1085
1086
1087            default: {
1088                //do nothing!
1089
1090                break;
1091            }
1092        }
1093    }
1094
1095
1096    if (this->game_phase == "build settlement") {
1097        if (
1098            (this->tile_type != TileType :: OCEAN) and
1099            (this->tile_type != TileType :: LAKE)
1100        ) {
1101            if (this->event_ptr->key.code == sf::Keyboard::B) {
1102                this->__buildSettlement();
1103            }
1104        }
1105    }
1106
1107
1108    else if (this->game_phase == "system management") {
1109        if (this->has_improvement) {
1110            // will be caught by this->tile_improvement_ptr->processEvent();
1111        }
1112
1113
1114        else if (not this->resource_assessed) {
1115            if (this->event_ptr->key.code == sf::Keyboard::A) {
1116                if (this->credits < RESOURCE_ASSESSMENT_COST) {
1117                    std::cout << "Cannot assess resource: insufficient credits (need "
1118                        << RESOURCE_ASSESSMENT_COST << " K)" << std::endl;
1119
1120                    this->__sendInsufficientCreditsMessage();
1121                }
1122
1123                else {
1124                    this->assess();
1125                    this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1126                    this->__sendTileStateMessage();
1127                    this->__sendGameStateRequest();
1128                }
1129            }
1130        }
1131
1132
1133        else if (
1134            (not this->decoration_cleared) and
1135            (this->tile_type != TileType :: OCEAN) and
1136            (this->tile_type != TileType :: LAKE)
1137        ) {
1138            if (this->event_ptr->key.code == sf::Keyboard::C) {
1139                int clear_cost = 0;
1140
1141                switch (this->tile_type) {
1142                    case (TileType :: FOREST): {
1143                        clear_cost = CLEAR_FOREST_COST;
1144
1145                        break;
1146                    }
1147
1148
1149                    case (TileType :: MOUNTAINS): {
1150                        clear_cost = CLEAR_MOUNTAINS_COST;
1151
1152                        break;
1153                    }
1154
1155
1156                    case (TileType :: PLAINS): {
1157                        clear_cost = CLEAR_PLAINS_COST;
1158
1159                        break;
1160                    }
1161
1162
1163                    default: {
1164                        // do nothing!
1165
1166                        break;
1167                    }
1168                }
1169
```

```
1170                    if (this->credits < clear_cost) {
1171                        std::cout « "Cannot clear tile: insufficient credits (need "
1172                            « clear_cost « " K)" « std::endl;
1173
1174                        this->__sendInsufficientCreditsMessage();
1175                    }
1176
1177                    else {
1178                        this->__clearDecoration();
1179                        this->__sendCreditsSpentMessage(clear_cost);
1180                        this->__sendTileStateMessage();
1181                        this->__sendGameStateRequest();
1182                    }
1183                }
1184            }
1185
1186
1187        else if (
1188            (this->decoration_cleared) or
1189            (this->tile_type == TileType :: OCEAN) or
1190            (this->tile_type == TileType :: LAKE)
1191        ) {
1192            if (this->event_ptr->key.code == sf::Keyboard::B) {
1193                this->__openBuildMenu();
1194            }
1195        }
1196    }
1197
1198    return;
1199 }   /* __handleKeyPressEvents() */
```

4.7.3.16 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
1214 {
1215    switch (this->event_ptr->mouseButton.button) {
1216        case (sf::Mouse::Left): {
1217            if (this->__isClicked()) {
1218                std::cout « "Tile (" « this->position_x « ", " «
1219                    this->position_y « ") was selected" « std::endl;
1220
1221                this->__setIsSelected(true);
1222
1223                this->__sendTileSelectedMessage();
1224                this->__sendTileStateMessage();
1225                this->__sendGameStateRequest();
1226            }
1227
1228            else {
1229                this->__setIsSelected(false);
1230            }
1231
1232            break;
1233        }
1234
1235
1236        case (sf::Mouse::Right): {
1237            this->__setIsSelected(false);
1238
1239            break;
1240        }
1241
1242
1243        default: {
1244            // do nothing!
1245
1246            break;
1247        }
1248    }
1249
1250    return;
1251 }   /* __handleMouseButtonEvents() */
```

4.7.3.17 __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.

```
858 {
859     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
860
861     double mouse_x = mouse_position.x;
862     double mouse_y = mouse_position.y;
863
864     double distance = sqrt(
865         pow(this->position_x - mouse_x, 2) +
866         pow(this->position_y - mouse_y, 2)
867     );
868
869     if (distance < this->minor_radius) {
870         return true;
871     }
872     else {
873         return false;
874     }
875 }   /* __isClicked() */
```

4.7.3.18 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

```
1266 {
1267     if (this->build_menu_open) {
1268         return;
1269     }
1270
1271     this->build_menu_open = true;
1272     this->assets_manager_ptr->getSound("build menu open")->play();
1273
1274     return;
1275 }   /* __openBuildMenu() */
```

4.7.3.19 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2012 {
2013     Message assess_neighbours_message;
2014
2015     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2016     assess_neighbours_message.subject = "assess neighbours";
2017
2018     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2019
2020     std::cout << "Assess neighbours message sent by " << this << std::endl;
2021
2022     return;
2023 }   /* __sendAssessNeighboursMessage() */
```

4.7.3.20 __sendCreditsSpentMessage()

```
void HexTile::__sendCreditsSpentMessage (
            int credits_spent )  [private]
```

Helper method to format and send a credits spent message.

Parameters

credits_spent	The number of credits that were spent.

```
2095 {
2096     Message credits_spent_message;
2097
2098     credits_spent_message.channel = GAME_CHANNEL;
2099     credits_spent_message.subject = "credits spent";
2100
2101     credits_spent_message.int_payload["credits spent"] = credits_spent;
2102
2103     this->message_hub_ptr->sendMessage(credits_spent_message);
2104
2105     std::cout « "Credits spent (" « credits_spent « ") message sent by " « this
2106         « std::endl;
2107     return;
2108 }  /* __sendCreditsSpentMessage() */
```

4.7.3.21 __sendGameStateRequest()

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

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

```
2038 {
2039     Message game_state_request;
2040
2041     game_state_request.channel = GAME_CHANNEL;
2042     game_state_request.subject = "state request";
2043
2044     this->message_hub_ptr->sendMessage(game_state_request);
2045
2046     std::cout « "Game state request message sent by " « this « std::endl;
2047     return;
2048 }  /* __sendGameStateRequest() */
```

4.7.3.22 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2123 {
2124     Message insufficient_credits_message;
2125
2126     insufficient_credits_message.channel = GAME_CHANNEL;
2127     insufficient_credits_message.subject = "insufficient credits";
2128
2129     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2130
2131     std::cout « "Insufficient credits message sent by " « this « std::endl;
2132
2133     return;
2134 }  /* __sendInsufficientCreditsMessage() */
```

4.7.3.23 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1647 {
1648     Message tile_selected_message;
1649
1650     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1651     tile_selected_message.subject = "tile selected";
1652
1653     this->message_hub_ptr->sendMessage(tile_selected_message);
1654
1655     return;
1656 } /* __sendTileSelectedMessage() */
```

4.7.3.24 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
1970 {
1971     Message tile_state_message;
1972
1973     tile_state_message.channel = TILE_STATE_CHANNEL;
1974     tile_state_message.subject = "tile state";
1975
1976
1977     //                     32 char x 17 line console "------------------------------\n";
1978     std::string console_string                = "     **** TILE INFO ****       \n";
1979     console_string                          += "                               \n";
1980
1981     console_string                          += this->__getTileCoordsSubstring();
1982     console_string                          += "                               \n";
1983
1984     console_string                          += this->__getTileTypeSubstring();
1985     console_string                          += this->__getTileResourceSubstring();
1986     console_string                          += this->__getTileImprovementSubstring();
1987     console_string                          += "                               \n";
1988
1989     console_string                          += this->__getTileOptionsSubstring();
1990
1991     tile_state_message.string_payload["console string"] = console_string;
1992
1993     this->message_hub_ptr->sendMessage(tile_state_message);
1994
1995     std::cout << "Tile state message sent by " << this << std::endl;
1996     return;
1997 } /* __sendTileStateMessage() */
```

4.7.3.25 __sendUpdateGamePhaseMessage()

```
void HexTile::__sendUpdateGamePhaseMessage (
            std::string game_phase )  [private]
```

Helper method to format and send update game phase message.

Parameters

game_phase	The updated game phase.

```
2065 {
2066     Message update_game_phase_message;
2067
2068     update_game_phase_message.channel = GAME_CHANNEL;
2069     update_game_phase_message.subject = "update game phase";
2070
2071     update_game_phase_message.string_payload["game phase"] = game_phase;
2072
2073     this->message_hub_ptr->sendMessage(update_game_phase_message);
2074
2075     std::cout << "Update game phase message sent by " << this << std::endl;
2076
2077     return;
2078 }   /* __sendUpdateGamePhaseMessage() */
```

4.7.3.26 __setIsSelected()

```
void HexTile::__setIsSelected (
            bool is_selected )  [private]
```

Helper method to set the is selected attribute (of tile and improvement).

Parameters

is_selected	The value to set the is selected attribute to.

```
763 {
764     this->is_selected = is_selected;
765
766     if (this->tile_improvement_ptr != NULL) {
767         this->tile_improvement_ptr->is_selected = is_selected;
768
769         if (is_selected) {
770             switch (this->tile_improvement_ptr->tile_improvement_type) {
771                 case (TileImprovementType :: SETTLEMENT): {
772                     this->assets_manager_ptr->getSound("people and children")->play();
773
774                     break;
775                 }
776
777
778                 default: {
779                     // do nothing!
780
781                     break;
782                 }
783             }
784         }
785     }
786
787     if ((not is_selected) and this->build_menu_open) {
788         this->__closeBuildMenu();
789     }
790
791     return;
792 }   /* __toggleIsSelected() */
```

4.7.3.27 __setResourceText()

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

Helper method to set up resource text.

```
193 {
194     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
195
```

```
196     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
197
198     if (this->resource_assessed) {
199         switch (this->tile_resource) {
200             case (TileResource :: POOR): {
201                 this->resource_text.setString("-2");
202                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
203
204                 break;
205             }
206
207             case (TileResource :: BELOW_AVERAGE): {
208                 this->resource_text.setString("-1");
209                 this->resource_text.setFillColor(MONOCHROME_TEXT_RED);
210
211                 break;
212             }
213
214             case (TileResource :: AVERAGE): {
215                 this->resource_text.setString("+0");
216
217                 break;
218             }
219
220             case (TileResource :: ABOVE_AVERAGE): {
221                 this->resource_text.setString("+1");
222                 this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
223
224                 break;
225             }
226
227             case (TileResource :: GOOD): {
228                 this->resource_text.setString("+2");
229                 this->resource_text.setFillColor(MONOCHROME_TEXT_GREEN);
230
231                 break;
232             }
233
234             default: {
235                 this->resource_text.setString("");
236
237                 break;
238             }
239         }
240     }
241
242     else {
243         this->resource_text.setString("");
244     }
245
246     this->resource_text.setCharacterSize(20);
247
248     this->resource_text.setOrigin(
249         this->resource_text.getLocalBounds().width / 2,
250         this->resource_text.getLocalBounds().height / 2
251     );
252
253     this->resource_text.setPosition(
254         this->position_x,
255         this->position_y - 4
256     );
257
258     this->resource_text.setOutlineThickness(1);
259     this->resource_text.setOutlineColor(sf::Color(0, 0, 0, 255));
260
261     return;
262 } /* __setResourceText() */
```

4.7.3.28 __setUpBuildMenu()

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

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

```
666 {
667     this->build_menu_options_vec.clear();
668     this->build_menu_options_text_vec.clear();
669
```

```
670     //  1. set up and place build menu backing and text
671     this->build_menu_backing.setSize(sf::Vector2f(600, 256));
672     this->build_menu_backing.setOrigin(300, 128);
673     this->build_menu_backing.setPosition(400, 400);
674     this->build_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
675     this->build_menu_backing.setOutlineColor(MENU_FRAME_GREY);
676     this->build_menu_backing.setOutlineThickness(4);
677
678     this->build_menu_backing_text.setString("**** BUILD MENU ****");
679     this->build_menu_backing_text.setFont(
680         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
681     );
682     this->build_menu_backing_text.setCharacterSize(16);
683     this->build_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
684     this->build_menu_backing_text.setOrigin(
685         this->build_menu_backing_text.getLocalBounds().width / 2, 0
686     );
687     this->build_menu_backing_text.setPosition(400, 400 - 128 + 4);
688
689     //  2. set up and place build menu option sprites and text
690     switch (this->tile_type) {
691         case (TileType :: FOREST): {
692             this->__setUpDieselGeneratorBuildOption();
693             this->__setUpSolarPVBuildOption();
694             this->__setUpWindTurbineBuildOption();
695             this->__setUpEnergyStorageSystemBuildOption();
696
697             break;
698         }
699
700
701         case (TileType :: LAKE): {
702             this->__setUpSolarPVBuildOption(true);
703             this->__setUpWindTurbineBuildOption(true);
704
705             break;
706         }
707
708
709         case (TileType :: MOUNTAINS): {
710             this->__setUpDieselGeneratorBuildOption();
711             this->__setUpSolarPVBuildOption();
712             this->__setUpWindTurbineBuildOption();
713             this->__setUpEnergyStorageSystemBuildOption();
714
715             break;
716         }
717
718
719         case (TileType :: OCEAN): {
720             this->__setUpWindTurbineBuildOption(false, true);
721             this->__setUpTidalTurbineBuildOption();
722             this->__setUpWaveEnergyConverterBuildOption();
723
724             break;
725         }
726
727
728         case (TileType :: PLAINS): {
729             this->__setUpDieselGeneratorBuildOption();
730             this->__setUpSolarPVBuildOption();
731             this->__setUpWindTurbineBuildOption();
732             this->__setUpEnergyStorageSystemBuildOption();
733
734             break;
735         }
736
737
738         default: {
739             // do nothing!
740
741             break;
742         }
743     }
744
745     return;
746 }   /* __setUpBuildMenu() */
```

4.7.3.29 __setUpBuildOption()

```
void HexTile::__setUpBuildOption (
```

```
              std::string texture_key,
              std::string option_string ) [private]
```

Helper method to set up and postion the sprite and text for a build option.

Parameters

texture_key	The key for the appropriate illustration asset for the build option.
option_string	A string for the build option.

```
357 {
358     size_t n_options = this->build_menu_options_vec.size();
359
360     //  1. set up option sprite(s)
361     this->build_menu_options_vec.push_back({});
362
363     if (not texture_key.empty()) {
364         sf::Sprite texture_sheet(
365             *(this->assets_manager_ptr->getTexture(texture_key))
366         );
367
368         int sheet_height = texture_sheet.getLocalBounds().height;
369         int n_subrects = sheet_height / 64;
370
371         for (int i = 0; i < n_subrects; i++) {
372             this->build_menu_options_vec.back().push_back(
373                 sf::Sprite(
374                     *(this->assets_manager_ptr->getTexture(texture_key)),
375                     sf::IntRect(0, i * 64, 64, 64)
376                 )
377             );
378
379             this->build_menu_options_vec.back().back().setOrigin(
380                 this->build_menu_options_vec.back().back().getLocalBounds().width / 2,
381                 this->build_menu_options_vec.back().back().getLocalBounds().height
382             );
383
384             this->build_menu_options_vec.back().back().setPosition(
385                 400 - 300 + 75 + n_options * 150,
386                 400 - 32
387             );
388         }
389     }
390
391     else {
392         this->build_menu_options_vec.back().push_back(sf::Sprite());
393     }
394
395
396     //  2. set up option text
397     this->build_menu_options_text_vec.push_back(
398         sf::Text(
399             option_string,
400             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
401             16
402         )
403     );
404
405     this->build_menu_options_text_vec.back().setOrigin(
406         this->build_menu_options_text_vec.back().getLocalBounds().width / 2,
407         0
408     );
409
410     this->build_menu_options_text_vec.back().setPosition(
411         400 - 300 + 75 + n_options * 150,
412         400 - 16 - 4
413     );
414
415     this->build_menu_options_text_vec.back().setFillColor(MONOCHROME_TEXT_GREEN);
416
417     return;
418 }   /* __setUpBuildOption() */
```

4.7.3.30 __setUpDieselGeneratorBuildOption()

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

Helper method to set up and position the diesel generator build option.

```
433 {
434     //  1. set up option sprite(s)
435     std::string texture_key = "diesel generator";
436
437     //  2. set up option string (up to 16 chars wide)
438     //                                    "----------------\n"
439     std::string diesel_generator_string = "DIESEL GENERATOR\n";
440     diesel_generator_string            += "                \n";
441     diesel_generator_string            += "CAPACITY: 100 kW\n";
442     diesel_generator_string            += "COST:      ";
443     diesel_generator_string            += std::to_string(DIESEL_GENERATOR_BUILD_COST);
444     diesel_generator_string            += " K\n\n\n";
445     diesel_generator_string            += "BUILD:    [D]   \n";
446
447     //  3. call general method
448     this->__setUpBuildOption(texture_key, diesel_generator_string);
449
450     return;
451 }   /* __setUpDieselGeneratorBuildOption() */
```

4.7.3.31 __setUpEnergyStorageSystemBuildOption()

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

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

```
633 {
634     //  1. set up option sprite(s)
635     std::string texture_key = "energy storage system";
636
637     //  2. set up option string (up to 16 chars wide)
638     //                                    "----------------\n"
639     std::string energy_storage_system_string   = " ENERGY STORAGE \n";
640     energy_storage_system_string              += "                \n";
641     energy_storage_system_string              += "CAPCTY:  500 kWh\n";
642     energy_storage_system_string              += "COST:      ";
643     energy_storage_system_string              += std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
644     energy_storage_system_string              += " K\n\n\n";
645     energy_storage_system_string              += "BUILD:    [E]   \n";
646
647     //  3. call general method
648     this->__setUpBuildOption(texture_key, energy_storage_system_string);
649
650     return;
651 }   /* __setUpEnergyStorageSystemBuildOption() */
```

4.7.3.32 __setUpMagnifyingGlassSprite()

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

Helper method to set up and position magnifying glass sprite.

```
277 {
278     this->magnifying_glass_sprite.setTexture(
279         *(this->assets_manager_ptr->getTexture("magnifying_glass_64x64_1"))
280     );
281
282     this->magnifying_glass_sprite.setOrigin(
283         this->magnifying_glass_sprite.getLocalBounds().width / 2,
284         this->magnifying_glass_sprite.getLocalBounds().height / 2
285     );
286
287     this->magnifying_glass_sprite.setPosition(
288         this->position_x,
289         this->position_y
290     );
291
292     return;
293 }   /* __setUpMagnifyingGlassSprite() */
```

4.7.3.33 __setUpNodeSprite()

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

Helper method to set up node sprite.

```
68  {
69      this->node_sprite.setRadius(4);
70
71      this->node_sprite.setOrigin(
72          this->node_sprite.getLocalBounds().width / 2,
73          this->node_sprite.getLocalBounds().height / 2
74      );
75
76      this->node_sprite.setPosition(this->position_x, this->position_y);
77
78      this->node_sprite.setFillColor(sf::Color(255, 0, 0, 255));
79
80      return;
81  }   /* __setUpNodeSprite() */
```

4.7.3.34 __setUpResourceChipSprite()

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

Helper method to set up resource chip sprite.

```
166 {
167     this->resource_chip_sprite.setRadius(2 * this->minor_radius / 3);
168
169     this->resource_chip_sprite.setOrigin(
170         this->resource_chip_sprite.getLocalBounds().width / 2,
171         this->resource_chip_sprite.getLocalBounds().height / 2
172     );
173
174     this->resource_chip_sprite.setPosition(this->position_x, this->position_y);
175
176     this->resource_chip_sprite.setFillColor(RESOURCE_CHIP_GREY);
177
178     return;
179 }   /* __setUpResourceChip() */
```

4.7.3.35 __setUpSelectOutlineSprite()

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

Helper method to set up select outline sprite.

```
130 {
131     int n_points = 6;
132
133     this->select_outline_sprite.setPointCount(n_points);
134
135     for (int i = 0; i < n_points; i++) {
136         this->select_outline_sprite.setPoint(
137             i,
138             sf::Vector2f(
139                 this->position_x + this->major_radius * cos((30 + 60 * i) * (M_PI / 180)),
140                 this->position_y + this->major_radius * sin((30 + 60 * i) * (M_PI / 180))
141             )
142         );
143     }
144
145     this->select_outline_sprite.setOutlineThickness(4);
146     this->select_outline_sprite.setOutlineColor(MONOCHROME_TEXT_RED);
147
148     this->select_outline_sprite.setFillColor(sf::Color(0, 0, 0, 0));
149
150     return;
151 }   /* __setUpSelectOutline() */
```

4.7.3.36 __setUpSolarPVBuildOption()

```
void HexTile::__setUpSolarPVBuildOption (
            bool is_lake = false ) [private]
```

Helper method to set up and position the solar PV array build option.

Parameters

is_lake	If being built on a lake.

```
521 {
522     //  1. set up option sprite(s)
523     std::string texture_key = "solar PV array";
524
525     //  2. set up option string (up to 16 chars wide)
526     int build_cost = SOLAR_PV_BUILD_COST;
527     if (is_lake) {
528         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
529     }
530
531     //                                "----------------\n"
532     std::string solar_PV_string       = " SOLAR PV ARRAY \n";
533     solar_PV_string                  += "                \n";
534     solar_PV_string                  += "CAPACITY: 100 kW\n";
535     solar_PV_string                  += "COST:      ";
536     solar_PV_string                  += std::to_string(build_cost);
537     solar_PV_string                  += " K";
538
539     if (is_lake) {
540         solar_PV_string += "\n** LAKE BUILD **\n\n";
541     }
542     else {
543         solar_PV_string += "\n\n\n";
544     }
545
546     solar_PV_string                  += "BUILD:    [S]   \n";
547
548     //  3. call general method
549     this->__setUpBuildOption(texture_key, solar_PV_string);
550
551     return;
552 }   /* __setUpSolarPVBuildOption() */
```

4.7.3.37 __setUpTidalTurbineBuildOption()

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

Helper method to set up and position the tidal turbine build option.

```
567 {
568     //  1. set up option sprite(s)
569     std::string texture_key = "tidal turbine";
570
571     //  2. set up option string (up to 16 chars wide)
572     //                                "----------------\n"
573     std::string tidal_turbine_string   = " TIDAL TURBINE  \n";
574     tidal_turbine_string              += "                \n";
575     tidal_turbine_string              += "CAPACITY: 100 kW\n";
576     tidal_turbine_string              += "COST:      ";
577     tidal_turbine_string              += std::to_string(TIDAL_TURBINE_BUILD_COST);
578     tidal_turbine_string              += " K\n\n\n";
579     tidal_turbine_string              += "BUILD:    [T]   \n";
580
581     //  3. call general method
582     this->__setUpBuildOption(texture_key, tidal_turbine_string);
583
584     return;
585 }   /* __setUpTidalTurbineBuildOption() */
```

4.7.3.38 __setUpTileExplosionReel()

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

Helper method to set up tile explosion sprite reel.

```
308 {
309     for (int i = 0; i < 4; i++) {
310         for (int j = 0; j < 4; j++) {
311             this->explosion_sprite_reel.push_back(
312                 sf::Sprite(
313                     *(this->assets_manager_ptr->getTexture("tile clear explosion")),
314                     sf::IntRect(j * 64, i * 64, 64, 64)
315                 )
316             );
317
318             this->explosion_sprite_reel.back().setOrigin(
319                 this->explosion_sprite_reel.back().getLocalBounds().width / 2,
320                 this->explosion_sprite_reel.back().getLocalBounds().height / 2
321             );
322
323             this->explosion_sprite_reel.back().setPosition(
324                 this->position_x,
325                 this->position_y
326             );
327         }
328     }
329
330     return;
331 }   /* __setUpTileExplosionReel() */
```

4.7.3.39 __setUpTileSprite()

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

Helper method to set up tile sprite.

```
96 {
97     int n_points = 6;
98
99     this->tile_sprite.setPointCount(n_points);
100
101     for (int i = 0; i < n_points; i++) {
102         this->tile_sprite.setPoint(
103             i,
104             sf::Vector2f(
105                 this->position_x + this->major_radius * cos((30 + 60 * i) * (M_PI / 180)),
106                 this->position_y + this->major_radius * sin((30 + 60 * i) * (M_PI / 180))
107             )
108         );
109     }
110
111     this->tile_sprite.setOutlineThickness(1);
112     this->tile_sprite.setOutlineColor(sf::Color(175, 175, 175, 255));
113
114     return;
115 }   /* __setUpTileSprite() */
```

4.7.3.40 __setUpWaveEnergyConverterBuildOption()

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

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

```
600 {
601     // 1. set up option sprite(s)
```

```
602     std::string texture_key = "wave energy converter";
603
604     // 2. set up option string (up to 16 chars wide)
605     //                                      "----------------\n"
606     std::string wave_energy_converter_string   = "WAVE ENERGY CVTR\n";
607     wave_energy_converter_string               += "                \n";
608     wave_energy_converter_string               += "CAPACITY: 100 kW\n";
609     wave_energy_converter_string               += "COST:       ";
610     wave_energy_converter_string               += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
611     wave_energy_converter_string               += " K\n\n\n";
612     wave_energy_converter_string               += "BUILD:    [A]   \n";
613
614     // 3. call general method
615     this->__setUpBuildOption(texture_key, wave_energy_converter_string);
616
617     return;
618 } /* __setUpWaveEnergyConverterBuildOption() */
```

4.7.3.41 __setUpWindTurbineBuildOption()

```
void HexTile::__setUpWindTurbineBuildOption (
            bool is_lake = false,
            bool is_ocean = false )  [private]
```

Helper method to set up and position the wind turbine build option.

Parameters

is_lake	If being built on a lake tile.
is_ocean	If being built on an ocean tile.

```
470 {
471     // 1. set up option sprite(s)
472     std::string texture_key = "wind turbine";
473
474     // 2. set up option string (up to 16 chars wide)
475     int build_cost = WIND_TURBINE_BUILD_COST;
476     if (is_lake or is_ocean) {
477         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
478     }
479
480     //                                      "----------------\n"
481     std::string wind_turbine_string    = "  WIND TURBINE  \n";
482     wind_turbine_string                += "                \n";
483     wind_turbine_string                += "CAPACITY: 100 kW\n";
484     wind_turbine_string                += "COST:      ";
485     wind_turbine_string                += std::to_string(build_cost);
486     wind_turbine_string                += " K";
487
488     if (is_lake) {
489         wind_turbine_string += "\n** LAKE BUILD **\n\n";
490     }
491     else if (is_ocean) {
492         wind_turbine_string += "\n* OCEAN BUILD * \n\n";
493     }
494     else {
495         wind_turbine_string += "\n\n\n";
496     }
497
498     wind_turbine_string                += "BUILD:    [W]   \n";
499
500     // 3. call general method
501     this->__setUpBuildOption(texture_key, wind_turbine_string);
502
503     return;
504 } /* __setUpWindTurbineBuildOption() */
```

4.7.3.42 assess()

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

Method to assess the tile's resource.

```
2555 {
2556      this->resource_assessed = true;
2557      this->resource_assessment = true;
2558
2559      this->assets_manager_ptr->getSound("resource assessment")->play();
2560
2561      this->__setResourceText();
2562      this->__sendTileStateMessage();
2563
2564      return;
2565 }   /* assess() */
```

4.7.3.43 decorateTile()

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

Method to decorate tile.

```
2433 {
2434      switch (this->tile_type) {
2435          case (TileType :: FOREST): {
2436              this->tile_decoration_sprite.setTexture(
2437                  *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2438              );
2439
2440              break;
2441          }
2442
2443          case (TileType :: LAKE): {
2444              this->tile_decoration_sprite.setTexture(
2445                  *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2446              );
2447
2448              break;
2449          }
2450
2451          case (TileType :: MOUNTAINS): {
2452              this->tile_decoration_sprite.setTexture(
2453                  *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2454              );
2455
2456              break;
2457          }
2458
2459          case (TileType :: OCEAN): {
2460              this->tile_decoration_sprite.setTexture(
2461                  *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2462              );
2463
2464              break;
2465          }
2466
2467          case (TileType :: PLAINS): {
2468              this->tile_decoration_sprite.setTexture(
2469                  *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2470              );
2471
2472              break;
2473          }
2474
2475          default: {
2476              // do nothing!
2477
2478              break;
2479          }
2480      }
2481
2482
2483      if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```
2484            this->tile_decoration_sprite.setOrigin(
2485                this->tile_decoration_sprite.getLocalBounds().width / 2,
2486                this->tile_decoration_sprite.getLocalBounds().height / 2
2487            );
2488
2489            this->tile_decoration_sprite.setPosition(
2490                this->position_x,
2491                this->position_y
2492            );
2493
2494            if ((double)rand() / RAND_MAX > 0.5) {
2495                this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2496            }
2497        }
2498
2499        else {
2500            this->tile_decoration_sprite.setOrigin(
2501                this->tile_decoration_sprite.getLocalBounds().width / 2,
2502                this->tile_decoration_sprite.getLocalBounds().height
2503            );
2504
2505            this->tile_decoration_sprite.setPosition(
2506                this->position_x,
2507                this->position_y + 12
2508            );
2509
2510            if ((double)rand() / RAND_MAX > 0.5) {
2511                this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2512            }
2513        }
2514
2515        return;
2516 }    /* decorateTile(void) */
```

4.7.3.44 draw()

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

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

```
2660 {
2661     //  1. draw hex
2662     this->render_window_ptr->draw(this->tile_sprite);
2663
2664     //  2. draw node
2665     if (this->show_node) {
2666         this->render_window_ptr->draw(this->node_sprite);
2667     }
2668
2669     //  3. draw tile decoration
2670     if (not this->decoration_cleared) {
2671         this->render_window_ptr->draw(this->tile_decoration_sprite);
2672     }
2673
2674     //  4. draw tile improvement
2675     if (this->has_improvement) {
2676         if (not this->tile_improvement_ptr->just_built) {
2677             this->tile_improvement_ptr->draw();
2678         }
2679     }
2680
2681     //  5. draw resource
2682     if (this->show_resource) {
2683         this->render_window_ptr->draw(this->resource_chip_sprite);
2684         this->render_window_ptr->draw(this->resource_text);
2685     }
2686
2687     //  6. draw selection outline
2688     if (this->is_selected) {
2689         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2690
2691         outline_colour.a =
2692             255 * pow(cos((M_PI * this->frame) / (1.5 * FRAMES_PER_SECOND)), 2);
2693
2694         this->select_outline_sprite.setOutlineColor(outline_colour);
2695
2696         this->render_window_ptr->draw(this->select_outline_sprite);
2697     }
```

```
2698
2699      //  7. draw resource assessment notification
2700      if (this->resource_assessment) {
2701          int alpha = this->magnifying_glass_sprite.getColor().a;
2702
2703          alpha -= 0.05 * FRAMES_PER_SECOND;
2704          if (alpha < 0) {
2705              alpha = 0;
2706              this->resource_assessment = false;
2707          }
2708
2709          this->magnifying_glass_sprite.setColor(
2710              sf::Color(255, 255, 255, alpha)
2711          );
2712
2713          this->render_window_ptr->draw(this->magnifying_glass_sprite);
2714      }
2715
2716      //  8. draw explosion, then settlement placement
2717      if (this->draw_explosion) {
2718          this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2719
2720          if (this->frame % (FRAMES_PER_SECOND / 10) == 0) {
2721              this->explosion_frame++;
2722          }
2723
2724          if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2725              this->draw_explosion = false;
2726          }
2727      }
2728
2729      else if (this->has_improvement) {
2730          if (this->tile_improvement_ptr->just_built) {
2731              this->tile_improvement_ptr->draw();
2732          }
2733      }
2734
2735      //  9. build menu
2736      if (this->build_menu_open) {
2737          this->render_window_ptr->draw(this->build_menu_backing);
2738          this->render_window_ptr->draw(this->build_menu_backing_text);
2739
2740          for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2741              for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2742                  this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2743              }
2744              this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2745          }
2746      }
2747
2748      this->frame++;
2749      return;
2750 }    /* draw() */
```

4.7.3.45 processEvent()

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

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

```
2580 {
2581      //  1. process TileImprovement events
2582      if (this->tile_improvement_ptr != NULL) {
2583          this->tile_improvement_ptr->processEvent();
2584      }
2585
2586      //  2. process HexTile events
2587      if (this->event_ptr->type == sf::Event::KeyPressed) {
2588          this->__handleKeyPressEvents();
2589      }
2590
2591      if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2592          this->__handleMouseButtonEvents();
2593      }
2594
2595      return;
2596 }    /* processEvent() */
```

4.7.3.46 processMessage()

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

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

```
2611 {
2612     //  1. process TileImprovement messages
2613     if (this->tile_improvement_ptr != NULL) {
2614         this->tile_improvement_ptr->processMessage();
2615     }
2616
2617     //  2. process HexTile messages
2618     if (this->is_selected) {
2619         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2620             Message game_state_message = this->message_hub_ptr->receiveMessage(
2621                 GAME_STATE_CHANNEL
2622             );
2623
2624             if (game_state_message.subject == "game state") {
2625                 this->credits = game_state_message.int_payload["credits"];
2626                 this->game_phase = game_state_message.string_payload["game phase"];
2627
2628                 if (this->tile_improvement_ptr != NULL) {
2629                     this->tile_improvement_ptr->credits = this->credits;
2630                     this->tile_improvement_ptr->game_phase = this->game_phase;
2631                 }
2632
2633                 std::cout « "Game state message received by " « this « std::endl;
2634                 this->__sendTileStateMessage();
2635                 this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
2636             }
2637         }
2638
2639         std::cout « "Current credits (HexTile): " « this->credits « " K" «
2640             std::endl;
2641     }
2642
2643     return;
2644 }   /* processMessage() */
```

4.7.3.47 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].

```
2382 {
2383     //  1. check input
2384     if (input_value < 0 or input_value > 1) {
2385         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
2386         error_str += "not in the closed interval [0, 1]";
2387
2388         #ifdef _WIN32
2389             std::cout « error_str « std::endl;
2390         #endif  /* _WIN32 */
2391
2392         throw std::runtime_error(error_str);
2393     }
2394
2395     //  2. convert input value to tile resource
2396     TileResource tile_resource;
2397
2398     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2399         tile_resource = TileResource :: POOR;
2400     }
```

```
2401       else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2402           tile_resource = TileResource :: BELOW_AVERAGE;
2403       }
2404       else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2405           tile_resource = TileResource :: AVERAGE;
2406       }
2407       else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2408           tile_resource = TileResource :: ABOVE_AVERAGE;
2409       }
2410       else {
2411           tile_resource = TileResource :: GOOD;
2412       }
2413
2414       //  3. call alternate method
2415       this->setTileResource(tile_resource);
2416
2417       return;
2418 }   /* setTileResource(double) */
```

4.7.3.48 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.

```
2360 {
2361       this->tile_resource = tile_resource;
2362       this->__setResourceText();
2363
2364       return;
2365 }   /* setTileResource(TileResource) */
```

4.7.3.49 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].

```
2310 {
2311       //  1. check input
2312       if (input_value < 0 or input_value > 1) {
2313           std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
2314           error_str += "not in the closed interval [0, 1]";
2315
2316           #ifdef _WIN32
2317               std::cout « error_str « std::endl;
2318           #endif  /* _WIN32 */
2319
2320           throw std::runtime_error(error_str);
2321       }
2322
2323       //  2. convert input value to tile type
```

```
2324      TileType tile_type;
2325
2326      if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2327          tile_type = TileType :: LAKE;
2328      }
2329      else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2330          tile_type = TileType :: PLAINS;
2331      }
2332      else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2333          tile_type = TileType :: FOREST;
2334      }
2335      else {
2336          tile_type = TileType :: MOUNTAINS;
2337      }
2338
2339      //  3. call alternate method
2340      this->setTileType(tile_type);
2341
2342      return;
2343 }   /* setTileType(double) */
```

4.7.3.50 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.

```
2249 {
2250      this->tile_type = tile_type;
2251
2252      switch (this->tile_type) {
2253          case (TileType :: FOREST): {
2254              this->tile_sprite.setFillColor(FOREST_GREEN);
2255
2256              break;
2257          }
2258
2259          case (TileType :: LAKE): {
2260              this->tile_sprite.setFillColor(LAKE_BLUE);
2261
2262              break;
2263          }
2264
2265          case (TileType :: MOUNTAINS): {
2266              this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2267
2268              break;
2269          }
2270
2271          case (TileType :: OCEAN): {
2272              this->tile_sprite.setFillColor(OCEAN_BLUE);
2273
2274              break;
2275          }
2276
2277          case (TileType :: PLAINS): {
2278              this->tile_sprite.setFillColor(PLAINS_YELLOW);
2279
2280              break;
2281          }
2282
2283          default: {
2284              // do nothing!
2285
2286              break;
2287          }
2288      }
2289
2290      this->__setUpBuildMenu();
2291
```

```
2292    return;
2293 }   /* setTileType(TileType) */
```

4.7.3.51 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.
```
2531 {
2532     if (this->show_resource) {
2533         this->show_resource = false;
2534     }
2535     else {
2536         this->show_resource = true;
2537     }
2538
2539     return;
2540 }   /* toggleResourceOverlay() */
```

4.7.4 Member Data Documentation

4.7.4.1 assets_manager_ptr

AssetsManager* HexTile::assets_manager_ptr [private]

A pointer to the assets manager.

4.7.4.2 build_menu_backing

sf::RectangleShape HexTile::build_menu_backing

A backing for the tile build menu.

4.7.4.3 build_menu_backing_text

sf::Text HexTile::build_menu_backing_text

A text label for the build menu.

4.7.4.4 build_menu_open

`bool HexTile::build_menu_open`

A boolean which indicates if the tile build menu is open.

4.7.4.5 build_menu_options_text_vec

`std::vector<sf::Text> HexTile::build_menu_options_text_vec`

A vector of text for the tile build options.

4.7.4.6 build_menu_options_vec

`std::vector<std::vector<sf::Sprite> > HexTile::build_menu_options_vec`

A vector of sprites for illustrating the tile build options.

4.7.4.7 credits

`int HexTile::credits`

The current balance of credits.

4.7.4.8 decoration_cleared

`bool HexTile::decoration_cleared`

A boolean which indicates if the tile decoration has been cleared.

4.7.4.9 draw_explosion

`bool HexTile::draw_explosion`

A boolean which indicates whether or not to draw a tile explosion.

4.7.4.10 event_ptr

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

A pointer to the event class.

4.7.4.11 explosion_frame

`size_t HexTile::explosion_frame`

The current frame of the explosion animation.

4.7.4.12 explosion_sprite_reel

`std::vector<sf::Sprite> HexTile::explosion_sprite_reel`

A reel of sprites for a tile explosion animation.

4.7.4.13 frame

`unsigned long long int HexTile::frame`

The current frame of this object.

4.7.4.14 game_phase

`std::string HexTile::game_phase`

The current phase of the game.

4.7.4.15 has_improvement

`bool HexTile::has_improvement`

A boolean which indicates if tile has improvement or not.

4.7.4.16 is_selected

`bool HexTile::is_selected`

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

4.7.4.17 magnifying_glass_sprite

`sf::Sprite HexTile::magnifying_glass_sprite`

A magnifying glass sprite.

4.7.4.18 major_radius

`double HexTile::major_radius`

The radius of the smallest bounding circle.

4.7.4.19 message_hub_ptr

`MessageHub* HexTile::message_hub_ptr [private]`

A pointer to the message hub.

4.7.4.20 minor_radius

`double HexTile::minor_radius`

The radius of the largest inscribed circle.

4.7.4.21 node_sprite

`sf::CircleShape HexTile::node_sprite`

A circle shape to mark the tile node.

4.7.4.22 position_x

`double HexTile::position_x`

The x position of the tile.

4.7.4.23 position_y

`double HexTile::position_y`

The y position of the tile.

4.7.4.24 render_window_ptr

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

A pointer to the render window.

4.7.4.25 resource_assessed

`bool HexTile::resource_assessed`

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

4.7.4.26 resource_assessment

`bool HexTile::resource_assessment`

A boolean which triggers a resource assessment notification.

4.7.4.27 resource_chip_sprite

`sf::CircleShape HexTile::resource_chip_sprite`

A circle shape which represents a resource chip.

4.7.4.28 resource_text

`sf::Text HexTile::resource_text`

A text representation of the resource.

4.7.4.29 select_outline_sprite

`sf::ConvexShape HexTile::select_outline_sprite`

A convex shape which outlines the tile when selected.

4.7.4.30 show_node

`bool HexTile::show_node`

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

4.7.4.31 show_resource

`bool HexTile::show_resource`

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

4.7.4.32 tile_decoration_sprite

`sf::Sprite HexTile::tile_decoration_sprite`

A tile decoration sprite.

4.7.4.33 tile_improvement_ptr

[TileImprovement](#)* HexTile::tile_improvement_ptr

A pointer to the improvement for this tile.

4.7.4.34 tile_resource

`TileResource` `HexTile::tile_resource`

4.7.4.35 tile_sprite

`sf::ConvexShape HexTile::tile_sprite`

A convex shape which represents the tile.

4.7.4.36 tile_type

`TileType` `HexTile::tile_type`

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

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

4.8 Message Struct Reference

A structure which defines a standard message format.

`#include <MessageHub.h>`

Public Attributes

- std::string channel = ""

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

  A string describing the message subject.
- std::map< std::string, bool > bool_payload = {}

  A boolean payload.
- std::map< std::string, int > int_payload = {}

  A vector payload.
- std::map< std::string, double > double_payload = {}

  A vector payload.
- std::map< std::string, std::string > string_payload = {}

  A string payload.

4.8.1 Detailed Description

A structure which defines a standard message format.

4.8.2 Member Data Documentation

4.8.2.1 bool_payload

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

A boolean payload.

4.8.2.2 channel

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

A string identifying the appropriate channel for this message.

4.8.2.3 double_payload

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

A vector payload.

4.8.2.4 int_payload

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

A vector payload.

4.8.2.5 string_payload

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

A string payload.

4.8.2.6 subject

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

A string describing the message subject.

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

- header/ESC_core/MessageHub.h

4.9 MessageHub Class Reference

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

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

Public Member Functions

- MessageHub (void)

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

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

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

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

    Method to send a message to the message map. Channels are implemented in a first in, first out manner (i.e. message queue).
- bool isEmpty (std::string)

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

    Method to receive the first message in the channel. Channels are implemented in a first in, first out manner (i.e. message queue).
- void popMessage (std::string)

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

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

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

    Destructor for the MessageHub class.

Private Attributes

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

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

4.9.1 Detailed Description

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

4.9.2 Constructor & Destructor Documentation

4.9.2.1 MessageHub()

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

Constructor for the [MessageHub](#) class.

```
78 {
79     //...
80
81     std::cout « "MessageHub constructed at " « this « std::endl;
82
83     return;
84 }   /* MessageHub() */
```

4.9.2.2 ∼MessageHub()

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

Destructor for the [MessageHub](#) class.

```
425 {
426     this->clear();
427
428     std::cout « "MessageHub at " « this « " destroyed" « std::endl;
429
430     return;
431 }   /* ~MessageHub() */
```

4.9.3 Member Function Documentation

4.9.3.1 addChannel()

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

Method to add channel to message map.

Parameters

channel	The key for the message channel being added.

```
129 {
130     //  1. check if channel is in map (if so, throw error)
131     if (this->message_map.count(channel) > 0) {
132         std::string error_str = "ERROR  MessageHub::addChannel()  channel ";
133         error_str += channel;
134         error_str += " is already in message map";
135
136         #ifdef _WIN32
137             std::cout « error_str « std::endl;
138         #endif  /* _WIN32 */
139
140         throw std::runtime_error(error_str);
141     }
142
143     //  2. add channel to map
144     this->message_map[channel] = {};
145
146     std::cout « "Channel " « channel « " added to message hub" « std::endl;
147
148     return;
149 }   /* addChannel() */
```

4.9.3.2   clear()

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

Method to clear the MessageHub.
```
405 {
406
407     this->clearMessages();
408     this->message_map.clear();
409
410     return;
411 }   /* clear() */
```

4.9.3.3   clearMessages()

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

Method to clear messages from the MessageHub.
```
379 {
380     std::map<std::string, std::list<Message»::iterator map_iter;
381     for (
382         map_iter = this->message_map.begin();
383         map_iter != this->message_map.end();
384         map_iter++
385     ) {
386         map_iter->second.clear();
387     }
388
389     return;
390 }   /* clearMessages() */
```

4.9.3.4 hasTraffic()

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

Method to determine if there remains any message traffic.

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

4.9.3.5 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.

```
244 {
245     //  1. check if channel is in map (if not, throw error)
246     if (this->message_map.count(channel) <= 0) {
247         std::string error_str = "ERROR  MessageHub::isEmpty()  channel ";
248         error_str += channel;
249         error_str += " is not in message map";
250
251         #ifdef _WIN32
252             std::cout « error_str « std::endl;
253         #endif  /* _WIN32 */
254
255         throw std::runtime_error(error_str);
256     }
257
258     if (this->message_map[channel].empty()) {
259         return true;
260     }
261     else {
262         return false;
263     }
264 }   /* isEmpty() */
```

4.9.3.6 popMessage()

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

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

Parameters

channel	The key for the message channel being popped.

```
333  {
334      //  1. check if channel is in map (if not, throw error)
335      if (this->message_map.count(channel) <= 0) {
336          std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
337          error_str += channel;
338          error_str += " is not in message map";
339
340          #ifdef _WIN32
341              std::cout « error_str « std::endl;
342          #endif  /* _WIN32 */
343
344          throw std::runtime_error(error_str);
345      }
346
347      //  2. check if channel is empty (if so, throw error)
348      if (this->message_map[channel].empty()) {
349          std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
350          error_str += channel;
351          error_str += " is empty";
352
353          #ifdef _WIN32
354              std::cout « error_str « std::endl;
355          #endif  /* _WIN32 */
356
357          throw std::runtime_error(error_str);
358      }
359
360      //  3. pop message
361      this->message_map[channel].pop_front();
362
363      return;
364  }  /* popMessage() */
```

4.9.3.7   receiveMessage()

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

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

Parameters

channel	The key for the message channel being received from.

Returns

The first message in the given channel.

```
284  {
285      //  1. check if channel is in map (if not, throw error)
286      if (this->message_map.count(channel) <= 0) {
287          std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
288          error_str += channel;
289          error_str += " is not in message map";
290
291          #ifdef _WIN32
292              std::cout « error_str « std::endl;
293          #endif  /* _WIN32 */
294
```

```
295            throw std::runtime_error(error_str);
296        }
297
298        //  2. check if channel is empty (if so, throw error)
299        if (this->message_map[channel].empty()) {
300            std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
301            error_str += channel;
302            error_str += " is empty";
303
304            #ifdef _WIN32
305                std::cout « error_str « std::endl;
306            #endif  /* _WIN32 */
307
308            throw std::runtime_error(error_str);
309        }
310
311        //  3. receive message
312        Message message = this->message_map[channel].front();
313
314        return message;
315 }    /* receiveMessage() */
```

4.9.3.8 removeChannel()

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

Method to remove channel from message map.

Parameters

channel	The key for the message channel being removed.

```
166 {
167        //  1. check if channel is in map (if not, throw error)
168        if (this->message_map.count(channel) <= 0) {
169            std::string error_str = "ERROR  MessageHub::removeChannel()  channel ";
170            error_str += channel;
171            error_str += " is not in message map";
172
173            #ifdef _WIN32
174                std::cout « error_str « std::endl;
175            #endif  /* _WIN32 */
176
177            throw std::runtime_error(error_str);
178        }
179
180        //  2. remove channel from map
181        this->message_map[channel].clear();
182        this->message_map.erase(channel);
183
184        std::cout « "Channel " « channel « " removed from message hub" « std::endl;
185
186        return;
187 }    /* removeChannel() */
```

4.9.3.9 sendMessage()

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

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

Parameters

message	The message to be sent.

```
205 {
206     //  1. check if channel is in map (if not, throw error)
207     std::string channel = message.channel;
208
209     if (this->message_map.count(channel) <= 0) {
210         std::string error_str = "ERROR  MessageHub::sendMessage()  channel ";
211         error_str += channel;
212         error_str += " is not in message map";
213
214         #ifdef _WIN32
215             std::cout « error_str « std::endl;
216         #endif  /* _WIN32 */
217
218         throw std::runtime_error(error_str);
219     }
220
221     //  2. send message to message map
222     this->message_map[channel].push_back(message);
223
224     return;
225 }  /* sendMessage() */
```

4.9.4 Member Data Documentation

4.9.4.1 message_map

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

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

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

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

4.10 Settlement Class Reference

A settlement class (child class of TileImprovement).

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

Inheritance diagram for Settlement:
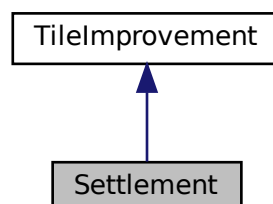
Collaboration diagram for Settlement:
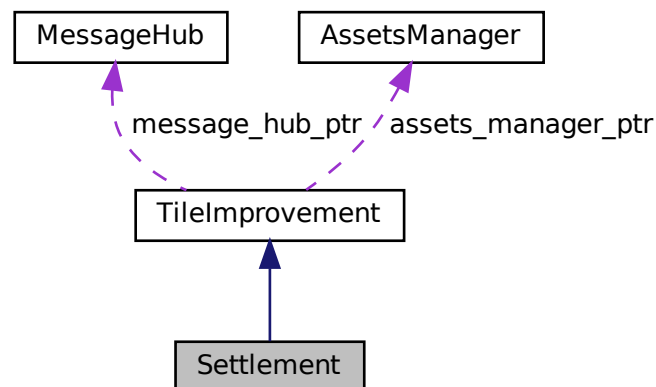
Public Member Functions

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

  Constructor for the Settlement class.
- void processEvent (void)

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

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

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

  Destructor for the Settlement class.

Public Attributes

- bool skip_smoke_processing

  A boolean which indicates whether or not to skip smoke processing.
- double smoke_da

  The per frame delta in smoke particle alpha value.
- double smoke_dx

  The per frame delta in smoke particle x position.
- double smoke_dy

  The per frame delta in smoke particle y position.
- double smoke_prob

  The probability of spawning a new smoke prob in any given frame.
- std::list< sf::Sprite > smoke_sprite_list

  A list of smoke sprite (for chimney animation).

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.10.1 Detailed Description

A settlement class (child class of TileImprovement).

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Settlement()

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

Constructor for the Settlement class.

Ref: Wikipedia [2023]

Parameters

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.

```
201   :
202 TileImprovement(
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
```

```
210 {
211     //  1. set attributes
212
213     //  1.1. private
214     //...
215
216     //  1.2. public
217     this->tile_improvement_type = TileImprovementType :: SETTLEMENT;
218
219     this->skip_smoke_processing = true;
220
221     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
222     this->smoke_dx = 5 * SECONDS_PER_FRAME;
223     this->smoke_dy = -10 * SECONDS_PER_FRAME;
224     this->smoke_prob = 2 * SECONDS_PER_FRAME;
225
226     this->smoke_sprite_list = {};
227
228     this->tile_improvement_string = "SETTLEMENT";
229
230     this->__setUpTileImprovementSpriteStatic();
231
232     std::cout « "Settlement constructed at " « this « std::endl;
233
234     return;
235 }   /* Settlement() */
```

4.10.2.2 ∼Settlement()

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

Destructor for the Settlement class.

```
387 {
388     std::cout « "Settlement at " « this « " destroyed" « std::endl;
389
390     return;
391 }   /* ~Settlement() */
```

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

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

4.10.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139
140         case (sf::Mouse::Right): {
141             //...
142
143             break;
144         }
145
146
147         default: {
148             // do nothing!
149
150             break;
151         }
152     }
153
154     return;
155 }   /* __handleMouseButtonEvents() */
```

4.10.3.3 __setUpTileImprovementSpriteStatic()

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

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

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

4.10.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
295 {
296      //  1. if just built, call base method and return
297      if (this->just_built) {
298          TileImprovement :: draw();
299
300          return;
301      }
302
303      //  2. draw static sprite and chimney smoke effects
304      this->render_window_ptr->draw(this->tile_improvement_sprite_static);
305
306      std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
307
308      double alpha = 255;
309
310      while (iter != this->smoke_sprite_list.end()) {
311          this->render_window_ptr->draw(*iter);
312
313          if (not this->skip_smoke_processing) {
314              alpha = (*iter).getColor().a;
315
316              alpha -= this->smoke_da;
317
318              if (alpha <= 0) {
319                  iter = this->smoke_sprite_list.erase(iter);
320                  continue;
321              }
322
323              (*iter).setColor(sf::Color(255, 255, 255, alpha));
324
325              (*iter).move(
326                  this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
327                  this->smoke_dy
328              );
329
330              (*iter).rotate(((double)rand() / RAND_MAX));
331          }
332
333          iter++;
334      }
335
336
337      if (not this->skip_smoke_processing) {
338          if ((double)rand() / RAND_MAX < smoke_prob) {
339              this->smoke_sprite_list.push_back(
340                  sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
341              );
342
343              this->smoke_sprite_list.back().setOrigin(
344                  this->smoke_sprite_list.back().getLocalBounds().width / 2,
345                  this->smoke_sprite_list.back().getLocalBounds().height / 2
346              );
347
348              this->smoke_sprite_list.back().setPosition(
349                  this->position_x + 9,
350                  this->position_y - 33
351              );
352          }
353      }
354
355
356      if (this->is_selected) {
357          if (this->skip_smoke_processing) {
358              this->skip_smoke_processing = false;
359          }
360
361          else {
362              this->skip_smoke_processing = true;
363          }
364      }
365
366      else {
367          this->skip_smoke_processing = false;
368      }
369
```

```
370     this->frame++;
371     return;
372 } /* draw() */
```

4.10.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
250 {
251     if (this->event_ptr->type == sf::Event::KeyPressed) {
252         this->__handleKeyPressEvents();
253     }
254
255     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
256         this->__handleMouseButtonEvents();
257     }
258
259     return;
260 } /* processEvent() */
```

4.10.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
275 {
276     //...
277
278     return;
279 } /* processMessage() */
```

4.10.4 Member Data Documentation

4.10.4.1 skip_smoke_processing

```
bool Settlement::skip_smoke_processing
```

A boolean which indicates whether or not to skip smoke processing.

4.10.4.2 smoke_da

```
double Settlement::smoke_da
```

The per frame delta in smoke particle alpha value.

4.10.4.3 smoke_dx

```
double Settlement::smoke_dx
```

The per frame delta in smoke particle x position.

4.10.4.4 smoke_dy

```
double Settlement::smoke_dy
```

The per frame delta in smoke particle y position.

4.10.4.5 smoke_prob

```
double Settlement::smoke_prob
```

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

4.10.4.6 smoke_sprite_list

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

A list of smoke sprite (for chimney animation).

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

- header/Settlement.h
- source/Settlement.cpp

4.11 SolarPV Class Reference

A settlement class (child class of TileImprovement).

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

Inheritance diagram for SolarPV:

```
┌─────────────────┐
│ TileImprovement │
└─────────────────┘
         ▲
         │
    ┌─────────┐
    │ SolarPV │
    └─────────┘
```

Collaboration diagram for SolarPV:
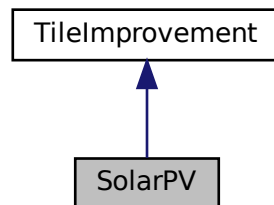
```
┌─────────────┐          ┌───────────────┐
│ MessageHub  │          │ AssetsManager │
└─────────────┘          └───────────────┘
        ▲                        ▲
         ╲  message_hub_ptr  assets_manager_ptr  ╱
          ╲                    ╱
         ┌─────────────────┐
         │ TileImprovement │
         └─────────────────┘
                  ▲
                  │
             ┌─────────┐
             │ SolarPV │
             └─────────┘
```
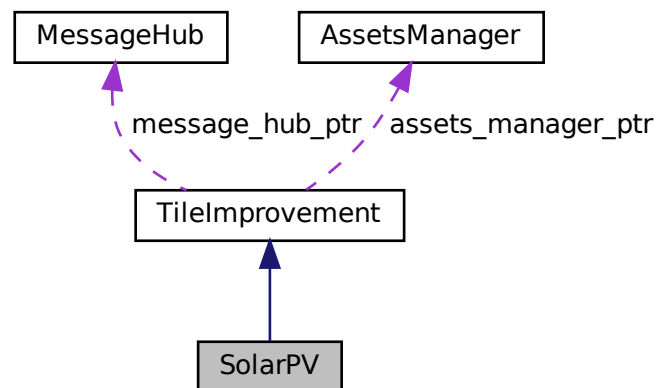
Public Member Functions

- **SolarPV** (double, double, sf::Event *, sf::RenderWindow *, AssetsManager *, MessageHub *)

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

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

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

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

    Destructor for the SolarPV class.

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

    Helper method to handle mouse button events.

Additional Inherited Members

4.11.1 Detailed Description

A settlement class (child class of TileImprovement).

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SolarPV()

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

Constructor for the SolarPV class.

Ref: Wikipedia [2023]

Parameters

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.

```
201   :
202 TileImprovement(
203     position_x,
204     position_y,
205     event_ptr,
206     render_window_ptr,
207     assets_manager_ptr,
208     message_hub_ptr
209 )
```

```
210 {
211     //  1. set attributes
212
213     //  1.1. private
214     //...
215
216     //  1.2. public
217     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
218
219     this->is_running = false;
220
221     this->tile_improvement_string = "SOLAR PV ARRAY";
222
223     this->__setUpTileImprovementSpriteStatic();
224
225     std::cout « "SolarPV constructed at " « this « std::endl;
226
227     return;
228 }    /* SolarPV() */
```

4.11.2.2 ∼SolarPV()

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

Destructor for the SolarPV class.

```
317 {
318     std::cout « "SolarPV at " « this « " destroyed" « std::endl;
319
320     return;
321 }    /* ~SolarPV() */
```

4.11.3 Member Function Documentation

4.11.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

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

4.11.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
131 {
132     switch (this->event_ptr->mouseButton.button) {
133         case (sf::Mouse::Left): {
134             //...
135
136             break;
137         }
138
139
140         case (sf::Mouse::Right): {
141             //...
142
143             break;
144         }
145
146
147         default: {
148             // do nothing!
149
150             break;
151         }
152     }
153
154     return;
155 }   /* __handleMouseButtonEvents() */
```

4.11.3.3 __setUpTileImprovementSpriteStatic()

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

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

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

4.11.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
288 {
289     //  1. if just built, call base method and return
290     if (this->just_built) {
291         TileImprovement :: draw();
292
293         return;
294     }
295
296
297     //  1. draw static sprite
298     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
299
300     this->frame++;
301     return;
302 }   /* draw() */
```

4.11.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
243 {
244     if (this->event_ptr->type == sf::Event::KeyPressed) {
245         this->__handleKeyPressEvents();
246     }
247
248     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
249         this->__handleMouseButtonEvents();
250     }
251
252     return;
253 }   /* processEvent() */
```

4.11.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
268 {
269     //...
270
271     return;
272 }   /* processMessage() */
```

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

- header/SolarPV.h
- source/SolarPV.cpp

4.12 TidalTurbine Class Reference

A settlement class (child class of TileImprovement).

`#include <TidalTurbine.h>`

Inheritance diagram for TidalTurbine:
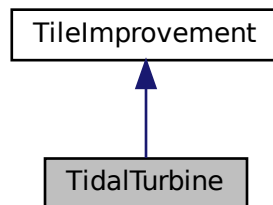
Collaboration diagram for TidalTurbine:
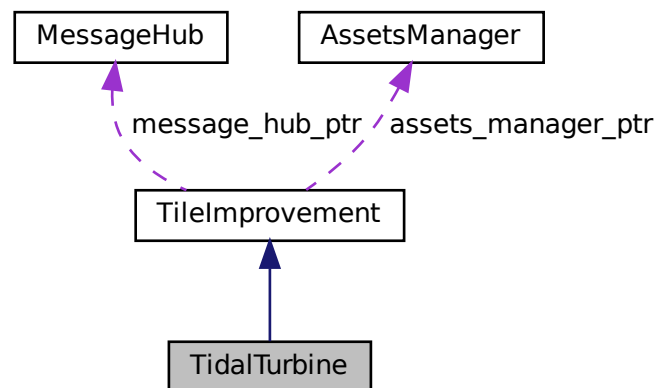
Public Member Functions

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

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

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

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

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

  Destructor for the TidalTurbine class.

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

    Helper method to handle mouse button events.

Additional Inherited Members

4.12.1 Detailed Description

A settlement class (child class of TileImprovement).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TidalTurbine()

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

Constructor for the TidalTurbine class.

Ref: Wikipedia [2023]

Parameters

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.

```
212  :
213 TileImprovement(
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
```

```
221 {
222     //  1. set attributes
223
224     //  1.1. private
225     //...
226
227     //  1.2. public
228     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
229
230     this->is_running = false;
231
232     this->tile_improvement_string = "TIDAL TURBINE";
233
234     this->__setUpTileImprovementSpriteAnimated();
235
236     std::cout « "TidalTurbine constructed at " « this « std::endl;
237
238     return;
239 }   /* TidalTurbine() */
```

4.12.2.2   ∼TidalTurbine()

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

Destructor for the TidalTurbine class.

```
340 {
341     std::cout « "TidalTurbine at " « this « " destroyed" « std::endl;
342
343     return;
344 }   /* ~TidalTurbine() */
```

4.12.3   Member Function Documentation

4.12.3.1   __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

```
114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118
119         default: {
120             // do nothing!
121
122             break;
123         }
124     }
125
126     return;
127 }   /* __handleKeyPressEvents() */
```

4.12.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146
147             break;
148         }
149
150
151         case (sf::Mouse::Right): {
152             //...
153
154             break;
155         }
156
157
158         default: {
159             // do nothing!
160
161             break;
162         }
163     }
164
165     return;
166 }   /* __handleMouseButtonEvents() */
```

4.12.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet(
70         *(this->assets_manager_ptr->getTexture("tidal turbine"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("tidal turbine")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 }   /* __setUpTileImprovementSpriteAnimated() */
```

4.12.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
299 {
300      //  1. if just built, call base method and return
301      if (this->just_built) {
302          TileImprovement :: draw();
303
304          return;
305      }
306
307
308      //  1. draw first element of animated sprite
309      this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
310
311
312      //  2. draw second element of animated sprite
313      if (this->is_running) {
314          //...
315      }
316
317      else {
318          //...
319      }
320
321      this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
322
323      this->frame++;
324      return;
325 }   /* draw() */
```

4.12.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
254 {
255      if (this->event_ptr->type == sf::Event::KeyPressed) {
256          this->__handleKeyPressEvents();
257      }
258
259      if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
260          this->__handleMouseButtonEvents();
261      }
262
263      return;
264 }   /* processEvent() */
```

4.12.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
279 {
280      //...
281
282      return;
283 }   /* processMessage() */
```

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

- header/TidalTurbine.h
- source/TidalTurbine.cpp

4.13 TileImprovement Class Reference

A base class for the tile improvement hierarchy.

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

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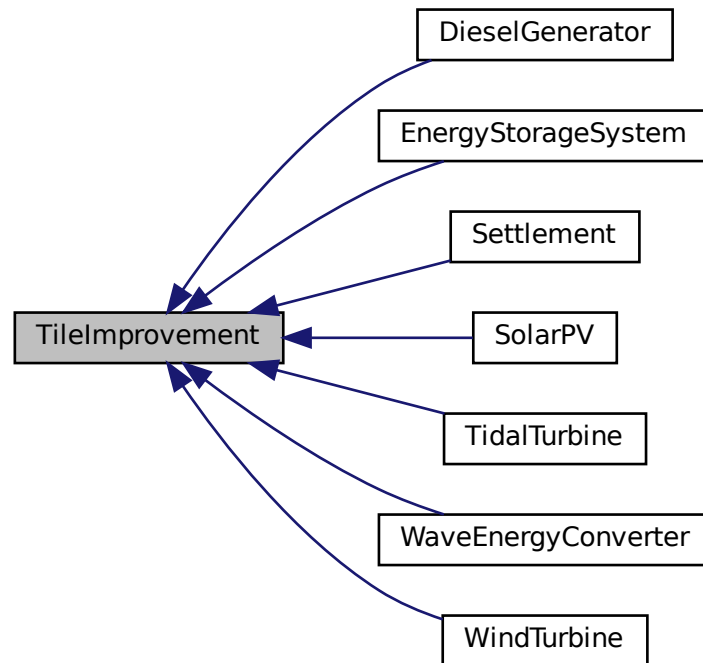
Collaboration diagram for TileImprovement:
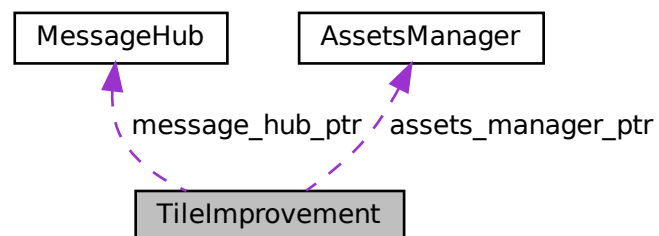
Public Member Functions

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

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

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

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

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

  Destructor for the TileImprovement class.

Public Attributes

- TileImprovementType tile_improvement_type

  The type of the tile improvement.
- bool is_running

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

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

  A boolean which indicates that the improvement was just built.
- unsigned long long int frame

  The current frame of this object.
- int credits

  The current balance of credits.
- double position_x

  The x position of the tile improvement.
- double position_y

  The y position of the tile improvement.
- std::string game_phase

  The current phase of the game.
- std::string tile_improvement_string

  A string representation of the tile improvement type.
- sf::Sprite tile_improvement_sprite_static

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

  An animated sprite, for the ContextMenu visual screen.

Protected Member Functions

- virtual void __handleKeyPressEvents (void)

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

  Helper method to handle mouse button events.

Protected Attributes

- sf::Event ∗ event_ptr

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

    A pointer to the render window.
- AssetsManager ∗ assets_manager_ptr

    A pointer to the assets manager.
- MessageHub ∗ message_hub_ptr

    A pointer to the message hub.

4.13.1 Detailed Description

A base class for the tile improvement hierarchy.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TileImprovement()

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

Constructor for the TileImprovement class.

Ref: Wikipedia [2023]

Parameters

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.

```
167 {
168     //  1. set attributes
169
170     //  1.1. protected
171     this->event_ptr = event_ptr;
172     this->render_window_ptr = render_window_ptr;
173
174     this->assets_manager_ptr = assets_manager_ptr;
175     this->message_hub_ptr = message_hub_ptr;
176
```

```
177     //  1.2. public
178     this->is_selected = true;
179     this->just_built = true;
180
181     this->frame = 0;
182     this->credits = 0;
183
184     this->position_x = position_x;
185     this->position_y = position_y;
186
187     this->game_phase = "build settlement";
188
189     std::cout « "TileImprovement constructed at " « this « std::endl;
190
191     return;
192 }   /* TileImprovement() */
```

4.13.2.2 ∼**TileImprovement()**

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

Destructor for the TileImprovement class.

```
381 {
382     std::cout « "TileImprovement at " « this « " destroyed" « std::endl;
383
384     return;
385 }   /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
68 {
69     switch (this->event_ptr->key.code) {
70         //...
71
72
73         default: {
74             // do nothing!
75
76             break;
77         }
78     }
79
80     return;
81 }   /* __handleKeyPressEvents() */
```

4.13.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
96  {
97      switch (this->event_ptr->mouseButton.button) {
98          case (sf::Mouse::Left): {
99              //...
100
101              break;
102          }
103
104
105          case (sf::Mouse::Right): {
106              //...
107
108              break;
109          }
110
111
112          default: {
113              // do nothing!
114
115              break;
116          }
117      }
118
119      return;
120  }   /* __handleMouseButtonEvents() */
```

4.13.3.3 draw()

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

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
252  {
253      if (this->tile_improvement_sprite_static.getTexture() != NULL) {
254          int alpha = this->tile_improvement_sprite_static.getColor().a;
255
256          alpha += 0.04 * FRAMES_PER_SECOND;
257
258          this->tile_improvement_sprite_static.setColor(
259              sf::Color(255, 255, 255, alpha)
260          );
261
262          this->tile_improvement_sprite_static.move(0, 25 * SECONDS_PER_FRAME);
263
264          if (
265              (alpha >= 255) or
266              (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
267          ) {
268              this->tile_improvement_sprite_static.setColor(
269                  sf::Color(255, 255, 255, 255)
270              );
271
272              this->tile_improvement_sprite_static.setPosition(
273                  this->position_x,
274                  this->position_y + 12
275              );
276
277              this->just_built = false;
278              this->assets_manager_ptr->getSound("place improvement")->play();
279          }
```

```
280
281            this->render_window_ptr->draw(this->tile_improvement_sprite_static);
282      }
283
284
285      else {
286          int alpha = 0;
287
288          for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
289              alpha = this->tile_improvement_sprite_animated[i].getColor().a;
290
291              alpha += 0.04 * FRAMES_PER_SECOND;
292
293              this->tile_improvement_sprite_animated[i].setColor(
294                  sf::Color(255, 255, 255, alpha)
295              );
296
297              this->tile_improvement_sprite_animated[i].move(0, 25 * SECONDS_PER_FRAME);
298
299              if (
300                  (alpha >= 255) or
301                  (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
302              ) {
303                  this->tile_improvement_sprite_animated[i].setColor(
304                      sf::Color(255, 255, 255, 255)
305                  );
306
307                  this->tile_improvement_sprite_animated[i].setPosition(
308                      this->position_x,
309                      this->position_y + 12
310                  );
311              }
312
313              this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
314          }
315
316          if (
317              (alpha >= 255) or
318              (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
319          ) {
320              this->just_built = false;
321              this->assets_manager_ptr->getSound("place improvement")->play();
322
323              switch (this->tile_improvement_type) {
324                  case (TileImprovementType :: WIND_TURBINE): {
325                      for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
326                          this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
327                          this->tile_improvement_sprite_animated[i].move(0, -32);
328                      }
329
330                      break;
331                  }
332
333
334                  case (TileImprovementType :: TIDAL_TURBINE): {
335                      for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
336                          this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
337                          this->tile_improvement_sprite_animated[i].move(0, -19);
338                      }
339
340                      break;
341                  }
342
343
344                  case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
345                      for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
346                          this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
347                          this->tile_improvement_sprite_animated[i].move(0, -32);
348                      }
349
350                      break;
351                  }
352
353
354                  default: {
355                      // do nothing!
356
357                      break;
358                  }
359              }
360          }
361      }
362
363
364      this->frame++;
365      return;
366  }   /* draw() */
```

4.13.3.4 processEvent()

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

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
207 {
208     if (this->event_ptr->type == sf::Event::KeyPressed) {
209         this->__handleKeyPressEvents();
210     }
211
212     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
213         this->__handleMouseButtonEvents();
214     }
215
216     return;
217 } /* processEvent() */
```

4.13.3.5 processMessage()

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

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
232 {
233     //...
234
235     return;
236 } /* processMessage() */
```

4.13.4 Member Data Documentation

4.13.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.13.4.2 credits

`int TileImprovement::credits`

The current balance of credits.

4.13.4.3 event_ptr

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

A pointer to the event class.

4.13.4.4 frame

`unsigned long long int TileImprovement::frame`

The current frame of this object.

4.13.4.5 game_phase

`std::string TileImprovement::game_phase`

The current phase of the game.

4.13.4.6 is_running

`bool TileImprovement::is_running`

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

4.13.4.7 is_selected

`bool TileImprovement::is_selected`

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

4.13.4.8 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.9 message_hub_ptr

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

A pointer to the message hub.

4.13.4.10 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.11 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.12 render_window_ptr

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

A pointer to the render window.

4.13.4.13 tile_improvement_sprite_animated

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

An animated sprite, for the ContextMenu visual screen.

4.13.4.14 tile_improvement_sprite_static

sf::Sprite TileImprovement::tile_improvement_sprite_static

A static sprite, for decorating the tile.

4.13.4.15 tile_improvement_string

std::string TileImprovement::tile_improvement_string

A string representation of the tile improvement type.

4.13.4.16 tile_improvement_type

TileImprovementType TileImprovement::tile_improvement_type

The type of the tile improvement.

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

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

4.14 WaveEnergyConverter Class Reference

A settlement class (child class of TileImprovement).

#include <WaveEnergyConverter.h>

Inheritance diagram for WaveEnergyConverter:
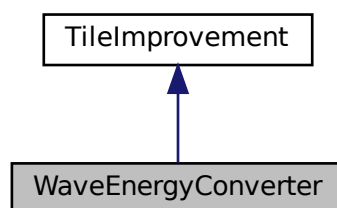
Collaboration diagram for WaveEnergyConverter:
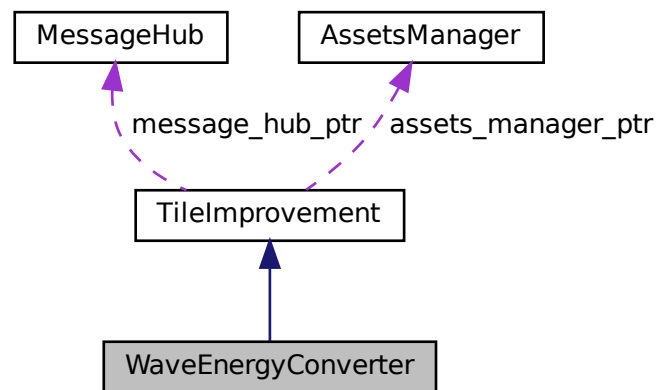
Public Member Functions

- WaveEnergyConverter (double, double, sf::Event *, sf::RenderWindow *, AssetsManager *, MessageHub *)

    Constructor for the WaveEnergyConverter class.
- void processEvent (void)

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

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

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

    Destructor for the WaveEnergyConverter class.

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

    Helper method to handle mouse button events.

Additional Inherited Members

4.14.1 Detailed Description

A settlement class (child class of TileImprovement).

4.14.2 Constructor & Destructor Documentation

4.14.2.1 WaveEnergyConverter()

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

Constructor for the WaveEnergyConverter class.

Ref: Wikipedia [2023]

Parameters

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.

```
212  :
213 TileImprovement(
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
221 {
222     //  1. set attributes
223
224     //  1.1. private
225     //...
226
227     //  1.2. public
228     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
229
230     this->is_running = false;
231
232     this->tile_improvement_string = "WAVE ENERGY";
233
234     this->__setUpTileImprovementSpriteAnimated();
235
236     std::cout « "WaveEnergyConverter constructed at " « this « std::endl;
237
238     return;
239 }  /* WaveEnergyConverter() */
```

4.14.2.2 ∼WaveEnergyConverter()

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

Destructor for the WaveEnergyConverter class.

```
340 {
341     std::cout « "WaveEnergyConverter at " « this « " destroyed" « std::endl;
342
343     return;
344 }   /* ~WaveEnergyConverter() */
```

4.14.3 Member Function Documentation

4.14.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

```
114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118
119         default: {
120             // do nothing!
121
122             break;
123         }
124     }
125
126     return;
127 }   /* __handleKeyPressEvents() */
```

4.14.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146
147             break;
148         }
149
150
151         case (sf::Mouse::Right): {
152             //...
153
154             break;
155         }
156
157
158         default: {
159             // do nothing!
160
161             break;
162         }
163     }
164
165     return;
166 }   /* __handleMouseButtonEvents() */
```

4.14.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet(
70         *(this->assets_manager_ptr->getTexture("wave energy converter"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("wave energy converter")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 }   /* __setUpTileImprovementSpriteAnimated() */
```

4.14.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
299 {
300     //  1. if just built, call base method and return
301     if (this->just_built) {
302         TileImprovement :: draw();
303
304         return;
305     }
306
307
308     //  1. draw first element of animated sprite
309     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
310
311
312     //  2. draw second element of animated sprite
313     if (this->is_running) {
314         //...
315     }
316
317     else {
318         //...
319     }
320
321     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
322
323     this->frame++;
324     return;
325 }   /* draw() */
```

4.14.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
254 {
255     if (this->event_ptr->type == sf::Event::KeyPressed) {
256         this->__handleKeyPressEvents();
257     }
258
259     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
260         this->__handleMouseButtonEvents();
261     }
262
263     return;
264 } /* processEvent() */
```

4.14.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
279 {
280     //...
281
282     return;
283 } /* processMessage() */
```

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

- header/WaveEnergyConverter.h
- source/WaveEnergyConverter.cpp

4.15 WindTurbine Class Reference

A settlement class (child class of TileImprovement).

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

Inheritance diagram for WindTurbine:
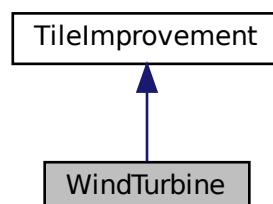
Collaboration diagram for WindTurbine:
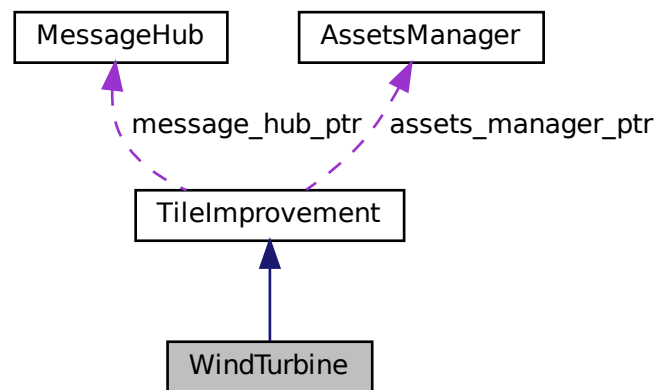
Public Member Functions

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

    Constructor for the WindTurbine class.
- void processEvent (void)

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

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

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

    Destructor for the WindTurbine class.

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

    Helper method to handle mouse button events.

Additional Inherited Members

4.15.1 Detailed Description

A settlement class (child class of TileImprovement).

4.15.2 Constructor & Destructor Documentation

4.15.2.1 WindTurbine()

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

Constructor for the WindTurbine class.

Ref: Wikipedia [2023]

Parameters

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.

```
212    :
213 TileImprovement(
214     position_x,
215     position_y,
216     event_ptr,
217     render_window_ptr,
218     assets_manager_ptr,
219     message_hub_ptr
220 )
221 {
222     //  1. set attributes
223
224     //  1.1. private
225     //...
226
227     //  1.2. public
228     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
229
230     this->is_running = false;
231
232     this->tile_improvement_string = "WIND TURBINE";
233
234     this->__setUpTileImprovementSpriteAnimated();
235
236     std::cout « "WindTurbine constructed at " « this « std::endl;
237
238     return;
239 }   /* WindTurbine() */
```

4.15.2.2 ∼WindTurbine()

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

Destructor for the WindTurbine class.

```
340 {
341     std::cout « "WindTurbine at " « this « " destroyed" « std::endl;
342
343     return;
344 }   /* ~WindTurbine() */
```

4.15.3 Member Function Documentation

4.15.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

Reimplemented from TileImprovement.

```
114 {
115     switch (this->event_ptr->key.code) {
116         //...
117
118
119         default: {
120             // do nothing!
121
122             break;
123         }
124     }
125
126     return;
127 }   /* __handleKeyPressEvents() */
```

4.15.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

Reimplemented from TileImprovement.

```
142 {
143     switch (this->event_ptr->mouseButton.button) {
144         case (sf::Mouse::Left): {
145             //...
146
147             break;
148         }
149
150
151         case (sf::Mouse::Right): {
152             //...
153
154             break;
155         }
156
157
158         default: {
159             // do nothing!
160
161             break;
162         }
163     }
164
165     return;
166 }   /* __handleMouseButtonEvents() */
```

4.15.3.3 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet(
70         *(this->assets_manager_ptr->getTexture("wind turbine"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("wind turbine")),
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97
98     return;
99 }   /* __setUpTileImprovementSpriteAnimated() */
```

4.15.3.4 draw()

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

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

Reimplemented from TileImprovement.

```
299 {
300     //  1. if just built, call base method and return
301     if (this->just_built) {
302         TileImprovement :: draw();
303
304         return;
305     }
306
307
308     //  1. draw first element of animated sprite
309     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
310
311
312     //  2. draw second element of animated sprite
313     if (this->is_running) {
314         //...
315     }
316
317     else {
318         //...
319     }
320
321     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
322
323     this->frame++;
324     return;
325 }   /* draw() */
```

4.15.3.5 processEvent()

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

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

Reimplemented from TileImprovement.

```
254 {
255     if (this->event_ptr->type == sf::Event::KeyPressed) {
256         this->__handleKeyPressEvents();
257     }
258
259     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
260         this->__handleMouseButtonEvents();
261     }
262
263     return;
264 } /* processEvent() */
```

4.15.3.6 processMessage()

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

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

Reimplemented from TileImprovement.

```
279 {
280     //...
281
282     return;
283 } /* processMessage() */
```

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

- header/WindTurbine.h
- source/WindTurbine.cpp

Chapter 5

File Documentation

5.1 header/ContextMenu.h File Reference

Header file for the ContextMenu class.

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

Include dependency graph for ContextMenu.h:
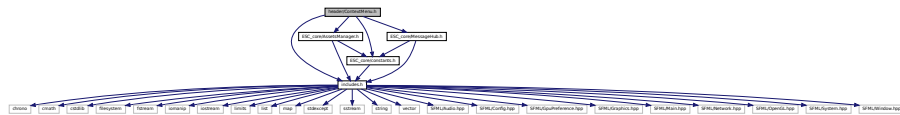
This graph shows which files directly or indirectly include this file:
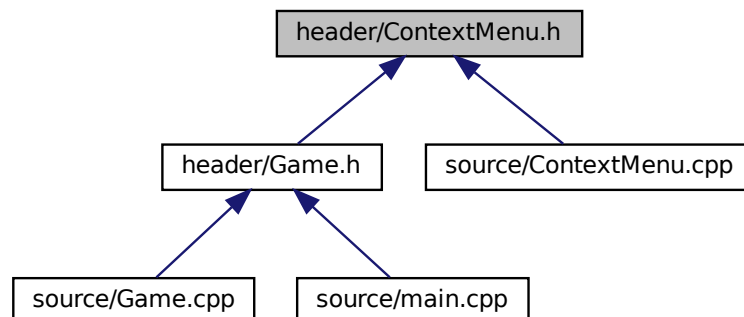
Classes

- class ContextMenu

    A class which defines a context menu for the game.

Enumerations

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

    An enumeration of the different console screen states.

5.1.1 Detailed Description

Header file for the ContextMenu class.

5.1.2 Enumeration Type Documentation

5.1.2.1 ConsoleState

enum ConsoleState

An enumeration of the different console screen states.

Enumerator

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

```
68              {
69     NONE_STATE,
70     READY,
71     MENU,
72     TILE,
73     N_CONSOLE_STATES
74 };
```

5.2 header/DieselGenerator.h File Reference

Header file for the DieselGenerator class.

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

```
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```
Include dependency graph for DieselGenerator.h:
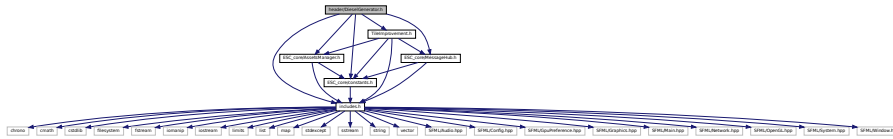
This graph shows which files directly or indirectly include this file:
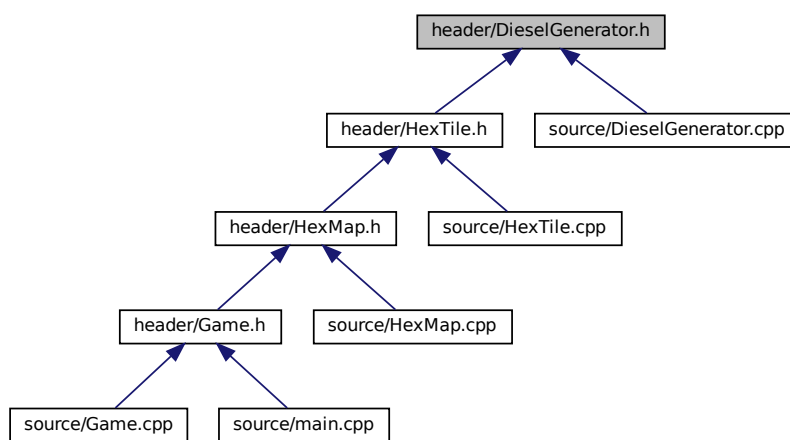
Classes

- class DieselGenerator

  A settlement class (child class of TileImprovement).

5.2.1 Detailed Description

Header file for the DieselGenerator class.

5.3 header/EnergyStorageSystem.h File Reference

Header file for the EnergyStorageSystem class.

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

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

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

Classes

- class EnergyStorageSystem

  A settlement class (child class of TileImprovement).

5.3.1 Detailed Description

Header file for the EnergyStorageSystem class.

5.4 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.

5.4.1 Detailed Description

Header file for the AssetsManager class.

5.5 header/ESC_core/constants.h File Reference

Header file for various constants.

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

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

Functions

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

    The base colour of a forest tile.
- const sf::Color LAKE_BLUE (0, 102, 204)

    The base colour of a lake (water) tile.
- const sf::Color MOUNTAINS_GREY (97, 110, 113)

    The base colour of a mountains tile.
- const sf::Color OCEAN_BLUE (0, 51, 102)

    The base colour of an ocean (water) tile.
- const sf::Color PLAINS_YELLOW (245, 222, 133)

    The base colour of a plains tile.
- const sf::Color RESOURCE_CHIP_GREY (175, 175, 175, 250)

    The base colour of the resource chip (backing).
- const sf::Color MENU_FRAME_GREY (185, 187, 182)

    The base colour of the context menu frame.
- const sf::Color MONOCHROME_SCREEN_BACKGROUND (40, 40, 40)

    The base colour of old monochrome screens.
- const sf::Color VISUAL_SCREEN_FRAME_GREY (151, 151, 143)

    The base colour of the framing of the visual screen.
- const sf::Color MONOCHROME_TEXT_GREEN (0, 255, 102)

    The base colour of old monochrome text (green).
- const sf::Color MONOCHROME_TEXT_AMBER (255, 176, 0)

    The base colour of old monochrome text (amber).
- const sf::Color MONOCHROME_TEXT_RED (255, 44, 0)

    The base colour of old monochrome text (red).

Variables

- const double FLOAT_TOLERANCE = 1e-6

    Tolerance for floating point equality tests.
- const unsigned long long int SECONDS_PER_YEAR = 31537970
- const unsigned long long int SECONDS_PER_MONTH = 2628164
- const int FRAMES_PER_SECOND = 60

    Target frames per second.
- const double SECONDS_PER_FRAME = 1.0 / 60

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

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

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

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

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

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

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

A message channel for tile state messages.
- const std::string HEX_MAP_CHANNEL = "HEX MAP CHANNEL"

  A message channel for hex map messages.
- const int CLEAR_FOREST_COST = 40

  The cost of clearing a forest tile.
- const int CLEAR_MOUNTAINS_COST = 250

  The cost of clearing a mountains tile.
- const int CLEAR_PLAINS_COST = 20

  The cost of clearing a plains tile.
- const int DIESEL_GENERATOR_BUILD_COST = 100

  The cost of building (or ugrading) a diesel generator.
- const int WIND_TURBINE_BUILD_COST = 400

  The cost of building (or upgrading) a wind turbine.
- const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25

  The additional cost of building on water.
- const int SOLAR_PV_BUILD_COST = 300

  The cost of building (or upgrading) a solar PV array.
- const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5

  The additional cost of building on water.
- const int TIDAL_TURBINE_BUILD_COST = 600

  The cost of building (or upgrading) a tidal turbine.
- const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800

  The cost of building (or upgrading) a wave energy converter.
- const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 400

  The cost of building (or upgrading) an energy storage system.
- const int STARTING_CREDITS = 500

  The starting balance of credits.
- const int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500

  The CO2-equivalent mass of emissions that would result from burning 1,000,000 L of diesel fuel.
- const int RESOURCE_ASSESSMENT_COST = 20

  The cost of doing a resource assessment.
- const int BUILD_SETTLEMENT_COST = 250

  The cost of building a settlement.
- const int STARTING_POPULATION = 100

  The starting population of a settlement.
- const double CO2E_KG_PER_LITRE_DIESEL = 3.1596

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

  A message channel for game messages.
- const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"

  A message channel for game state messages.

5.5.1 Detailed Description

Header file for various constants.

5.5.2 Function Documentation

5.5.2.1 FOREST_GREEN()

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

The base colour of a forest tile.

5.5.2.2 LAKE_BLUE()

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

The base colour of a lake (water) tile.

5.5.2.3 MENU_FRAME_GREY()

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

The base colour of the context menu frame.

5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()

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

The base colour of old monochrome screens.

5.5.2.5 MONOCHROME_TEXT_AMBER()

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

The base colour of old monochrome text (amber).

5.5.2.6 MONOCHROME_TEXT_GREEN()

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

The base colour of old monochrome text (green).

5.5.2.7 MONOCHROME_TEXT_RED()

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

The base colour of old monochrome text (red).

5.5.2.8 MOUNTAINS_GREY()

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

The base colour of a mountains tile.

5.5.2.9 OCEAN_BLUE()

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

The base colour of an ocean (water) tile.

5.5.2.10 PLAINS_YELLOW()

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

The base colour of a plains tile.

5.5.2.11 RESOURCE_CHIP_GREY()

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

The base colour of the resource chip (backing).

5.5.2.12 VISUAL_SCREEN_FRAME_GREY()

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

The base colour of the framing of the visual screen.

5.5.3 Variable Documentation

5.5.3.1 BUILD_SETTLEMENT_COST

```
const int BUILD_SETTLEMENT_COST = 250
```

The cost of building a settlement.

5.5.3.2 CLEAR_FOREST_COST

```
const int CLEAR_FOREST_COST = 40
```

The cost of clearing a forest tile.

5.5.3.3 CLEAR_MOUNTAINS_COST

```
const int CLEAR_MOUNTAINS_COST = 250
```

The cost of clearing a mountains tile.

5.5.3.4 CLEAR_PLAINS_COST

```
const int CLEAR_PLAINS_COST = 20
```

The cost of clearing a plains tile.

5.5.3.5 CO2E_KG_PER_LITRE_DIESEL

```
const double CO2E_KG_PER_LITRE_DIESEL = 3.1596
```

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

5.5.3.6 DIESEL_GENERATOR_BUILD_COST

```
const int DIESEL_GENERATOR_BUILD_COST = 100
```

The cost of building (or ugrading) a diesel generator.

5.5.3.7 EMISSIONS_LIFETIME_LIMIT_TONNES

```
const int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500
```

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

5.5.3.8 ENERGY_STORAGE_SYSTEM_BUILD_COST

```
const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 400
```

The cost of building (or upgrading) an energy storage system.

5.5.3.9 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

5.5.3.10 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

5.5.3.11 GAME_CHANNEL

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

A message channel for game messages.

5.5.3.12 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

5.5.3.13 GAME_STATE_CHANNEL

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

A message channel for game state messages.

5.5.3.14 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.15 HEX_MAP_CHANNEL

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

A message channel for hex map messages.

5.5.3.16 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

5.5.3.17 RESOURCE_ASSESSMENT_COST

const int RESOURCE_ASSESSMENT_COST = 20

The cost of doing a resource assessment.

5.5.3.18 SECONDS_PER_FRAME

const double SECONDS_PER_FRAME = 1.0 / 60

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

5.5.3.19 SECONDS_PER_MONTH

const unsigned long long int SECONDS_PER_MONTH = 2628164

5.5.3.20 SECONDS_PER_YEAR

const unsigned long long int SECONDS_PER_YEAR = 31537970

5.5.3.21 SOLAR_PV_BUILD_COST

const int SOLAR_PV_BUILD_COST = 300

The cost of building (or upgrading) a solar PV array.

5.5.3.22 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.23 STARTING_CREDITS

```
const int STARTING_CREDITS = 500
```

The starting balance of credits.

5.5.3.24 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.25 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

The cost of building (or upgrading) a tidal turbine.

5.5.3.26 TILE_RESOURCE_CUMULATIVE_PROBABILITIES

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

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

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

5.5.3.27 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.28 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.29 TILE_TYPE_CUMULATIVE_PROBABILITIES

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

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

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

5.5.3.30 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800
```

The cost of building (or upgrading) a wave energy converter.

5.5.3.31 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

The cost of building (or upgrading) a wind turbine.

5.5.3.32 WIND_TURBINE_WATER_BUILD_MULTIPLIER

```
const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25
```

The additional cost of building on water.

5.6 header/ESC_core/doxygen_cite.h File Reference

Header file which simply cites the doxygen tool.

5.6.1 Detailed Description

Header file which simply cites the doxygen tool.

Ref: van Heesch. [2023]

5.7 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:

5.7.1 Detailed Description

Header file for various includes.

Ref: Gomila [2023]

5.8 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:
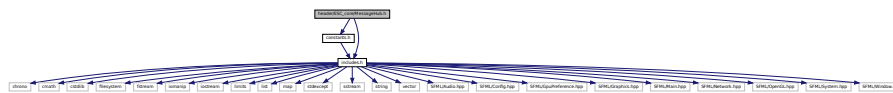
This graph shows which files directly or indirectly include this file:
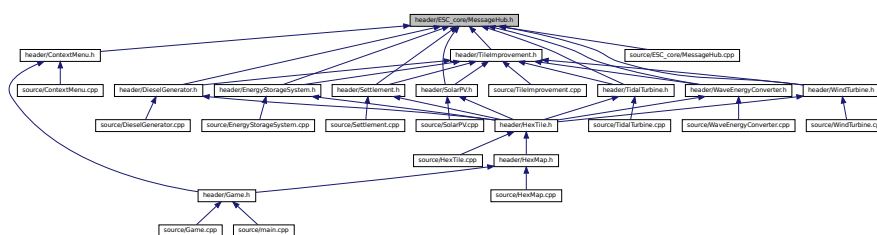
Classes

- 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.

5.8.1 Detailed Description

Header file for the MessageHub class.

5.9   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:
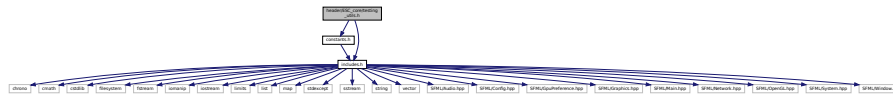
This graph shows which files directly or indirectly include this file:
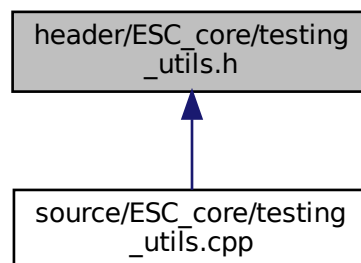
Functions

- 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.

5.9.1 Detailed Description

Header file for various testing utilities.

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

5.9.2 Function Documentation

5.9.2.1 expectedErrorNotDetected()

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

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

Parameters

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__").

```
462 {
463     std::string error_str = "\n ERROR  failed to throw expected error prior to line ";
464     error_str += std::to_string(line);
465     error_str += " of ";
466     error_str += file;
467
468     #ifdef _WIN32
469         std::cout « error_str « std::endl;
470     #endif
471
472     throw std::runtime_error(error_str);
473     return;
474 } /* expectedErrorNotDetected() */
```

5.9.2.2 printGold()

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

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

Parameters

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

```
114 {
115     std::cout « "\x1B[33m" « input_str « "\033[0m";
116     return;
117 } /* printGold() */
```

5.9.2.3 printGreen()

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

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

Parameters

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

```
94 {
95     std::cout « "\x1B[32m" « input_str « "\033[0m";
96     return;
97 }   /* printGreen() */
```

5.9.2.4 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.

```
134 {
135     std::cout « "\x1B[31m" « input_str « "\033[0m";
136     return;
137 }   /* printRed() */
```

5.9.2.5 testFloatEquals()

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

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

Parameters

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__").

```
168 {
169     if (fabs(x - y) <= FLOAT_TOLERANCE) {
170         return;
```

```
171     }
172
173     std::string error_str = "ERROR: testFloatEquals():\t in ";
174     error_str += file;
175     error_str += "\tline ";
176     error_str += std::to_string(line);
177     error_str += ":\t\n";
178     error_str += std::to_string(x);
179     error_str += " and ";
180     error_str += std::to_string(y);
181     error_str += " are not equal to within +/- ";
182     error_str += std::to_string(FLOAT_TOLERANCE);
183     error_str += "\n";
184
185     #ifdef _WIN32
186         std::cout « error_str « std::endl;
187     #endif
188
189     throw std::runtime_error(error_str);
190     return;
191 }   /* testFloatEquals() */
```

5.9.2.6 testGreaterThan()

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

Tests if x > y.

Parameters

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__").

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

5.9.2.7 testGreaterThanOrEqualTo()

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

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

Tests if x >= 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__").

```
272 {
273     if (x >= y) {
274         return;
275     }
276
277     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
278     error_str += file;
279     error_str += "\tline ";
280     error_str += std::to_string(line);
281     error_str += ":\t\n";
282     error_str += std::to_string(x);
283     error_str += " is not greater than or equal to ";
284     error_str += std::to_string(y);
285     error_str += "\n";
286
287     #ifdef _WIN32
288         std::cout << error_str << std::endl;
289     #endif
290
291     throw std::runtime_error(error_str);
292     return;
293 }   /* testGreaterThanOrEqualTo() */
```

5.9.2.8 testLessThan()

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

Tests if x < y.

Parameters

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__").

```
323 {
324     if (x < y) {
325         return;
326     }
327
328     std::string error_str = "ERROR: testLessThan():\t in ";
329     error_str += file;
330     error_str += "\tline ";
331     error_str += std::to_string(line);
332     error_str += ":\t\n";
```

```
333        error_str += std::to_string(x);
334        error_str += " is not less than ";
335        error_str += std::to_string(y);
336        error_str += "\n";
337
338        #ifdef _WIN32
339            std::cout << error_str << std::endl;
340        #endif
341
342        throw std::runtime_error(error_str);
343        return;
344    }   /* testLessThan() */
```

5.9.2.9   testLessThanOrEqualTo()

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

Tests if x <= 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__").

```
374 {
375        if (x <= y) {
376            return;
377        }
378
379        std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
380        error_str += file;
381        error_str += "\tline ";
382        error_str += std::to_string(line);
383        error_str += ":\t\n";
384        error_str += std::to_string(x);
385        error_str += " is not less than or equal to ";
386        error_str += std::to_string(y);
387        error_str += "\n";
388
389        #ifdef _WIN32
390            std::cout << error_str << std::endl;
391        #endif
392
393        throw std::runtime_error(error_str);
394        return;
395 }   /* testLessThanOrEqualTo() */
```

5.9.2.10   testTruth()

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

Tests if the given statement is true.

Parameters

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__").

```
422 {
423     if (statement) {
424         return;
425     }
426
427     std::string error_str = "ERROR: testTruth():\t in ";
428     error_str += file;
429     error_str += "\tline ";
430     error_str += std::to_string(line);
431     error_str += ":\t\n";
432     error_str += "Given statement is not true";
433
434     #ifdef _WIN32
435         std::cout « error_str « std::endl;
436     #endif
437
438     throw std::runtime_error(error_str);
439     return;
440 }   /* testTruth() */
```

5.10 header/Game.h File Reference

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

Include dependency graph for Game.h:
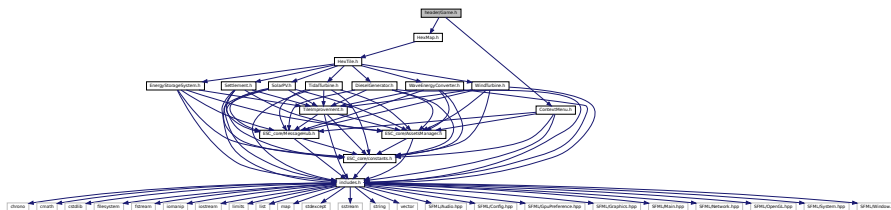
This graph shows which files directly or indirectly include this file:
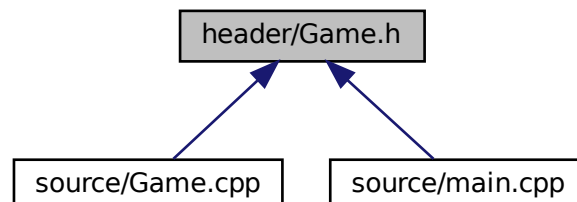
Classes

- class Game

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

Enumerations

- enum GamePhase {
  BUILD_SETTLEMENT , SYSTEM_MANAGEMENT , LOSS_EMISSIONS , LOSS_DEMAND ,
  LOSS_CREDITS , VICTORY , N_GAME_PHASES }

  An enumeration of the various game phases.

5.10.1 Enumeration Type Documentation

5.10.1.1 GamePhase

enum GamePhase

An enumeration of the various game phases.

Enumerator

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

```
66          {
67      BUILD_SETTLEMENT,
68      SYSTEM_MANAGEMENT,
69      LOSS_EMISSIONS,
70      LOSS_DEMAND,
71      LOSS_CREDITS,
72      VICTORY,
73      N_GAME_PHASES
74 }; /* GamePhase */
```

5.11 header/HexMap.h File Reference

Header file for the HexMap class.

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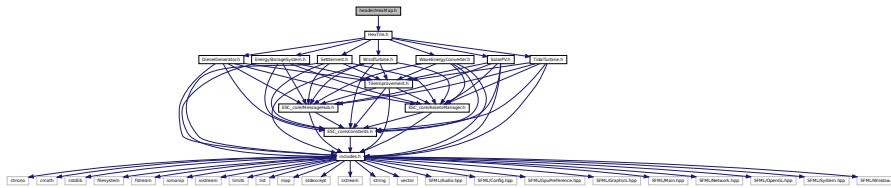
This graph shows which files directly or indirectly include this file:
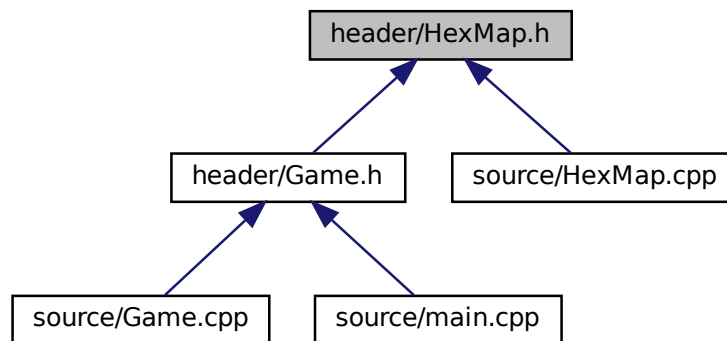
Classes

- class HexMap

    A class which defines a hex map of hex tiles.

5.11.1 Detailed Description

Header file for the HexMap class.

5.12 header/HexTile.h File Reference

Header file for the Game class.

```
#include "DieselGenerator.h"
#include "EnergyStorageSystem.h"
#include "Settlement.h"
#include "SolarPV.h"
#include "TidalTurbine.h"
#include "WaveEnergyConverter.h"
```

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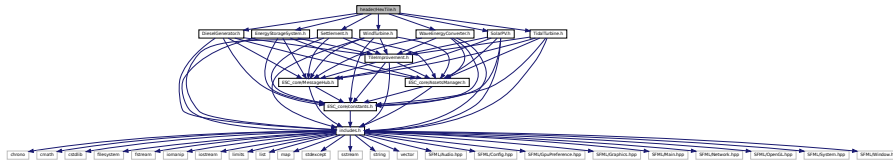
This graph shows which files directly or indirectly include this file:
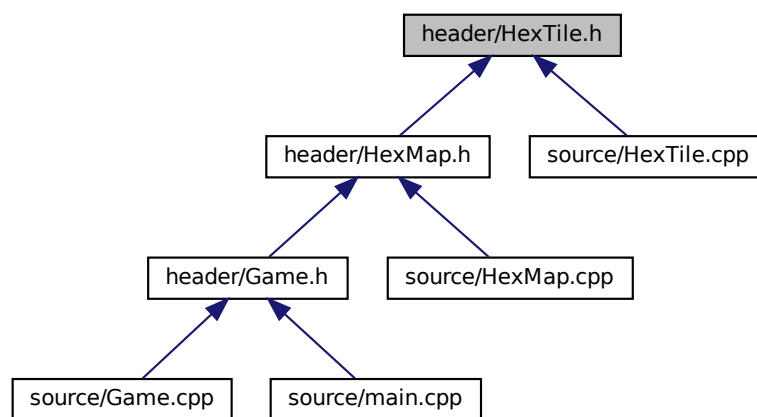
Classes

- class HexTile

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

Enumerations

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

  An enumeration of the different tile types.

- enum TileResource {
  POOR , BELOW_AVERAGE , AVERAGE , ABOVE_AVERAGE ,
  GOOD , N_TILE_RESOURCES }

  An enumeration of the different tile resource values.

5.12.1 Detailed Description

Header file for the Game class.

Header file for the HexTile class.

5.12.2 Enumeration Type Documentation

5.12.2.1 TileResource

```
enum TileResource
```

An enumeration of the different tile resource values.

Enumerator

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

```
88                 {
89     POOR,
90     BELOW_AVERAGE,
91     AVERAGE,
92     ABOVE_AVERAGE,
93     GOOD,
94     N_TILE_RESOURCES
95 }; /* TileResource */
```

5.12.2.2 TileType

```
enum TileType
```

An enumeration of the different tile types.

Enumerator

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

```
71                 {
72     NONE_TYPE,
73     FOREST,
74     LAKE,
75     MOUNTAINS,
76     OCEAN,
77     PLAINS,
78     N_TILE_TYPES
79 }; /* TileType */
```

5.13 header/Settlement.h File Reference

Header file for the Settlement class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```
Include dependency graph for Settlement.h:
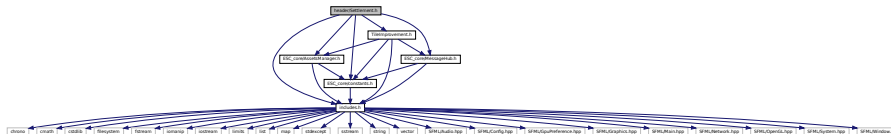
This graph shows which files directly or indirectly include this file:
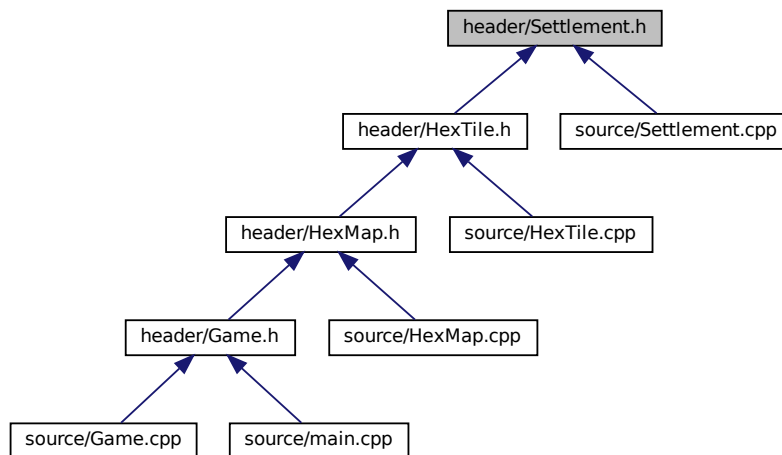
Classes

- class Settlement

    A settlement class (child class of TileImprovement).

5.13.1 Detailed Description

Header file for the Settlement class.

5.14 header/SolarPV.h File Reference

Header file for the SolarPV class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```
Include dependency graph for SolarPV.h:
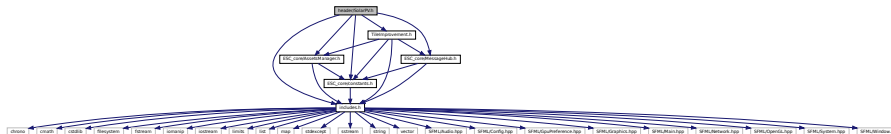
This graph shows which files directly or indirectly include this file:
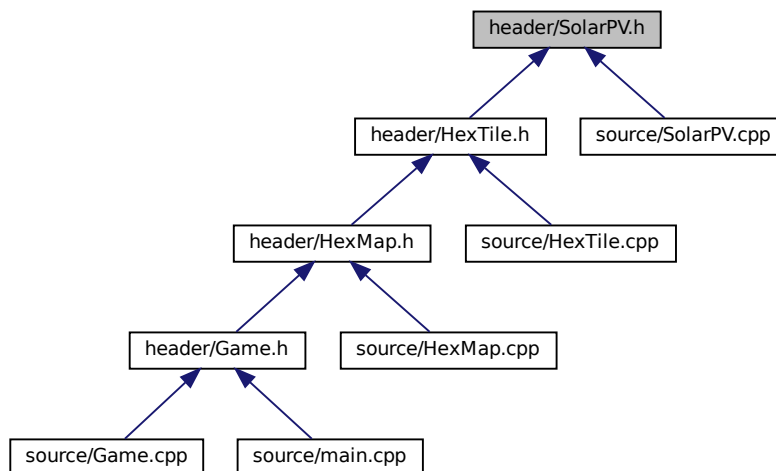
Classes

- class SolarPV

  A settlement class (child class of TileImprovement).

5.14.1 Detailed Description

Header file for the SolarPV class.

5.15 header/TidalTurbine.h File Reference

Header file for the TidalTurbine class.

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

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

Classes

- class TidalTurbine

    A settlement class (child class of TileImprovement).

5.15.1 Detailed Description

Header file for the TidalTurbine class.

5.16 header/TileImprovement.h File Reference

Header file for the TileImprovement class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```
Include dependency graph for TileImprovement.h:
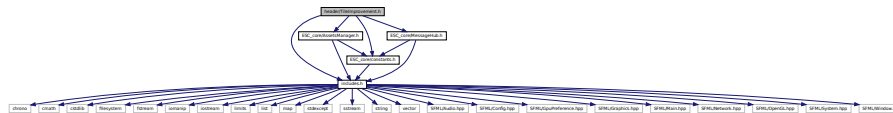
This graph shows which files directly or indirectly include this file:
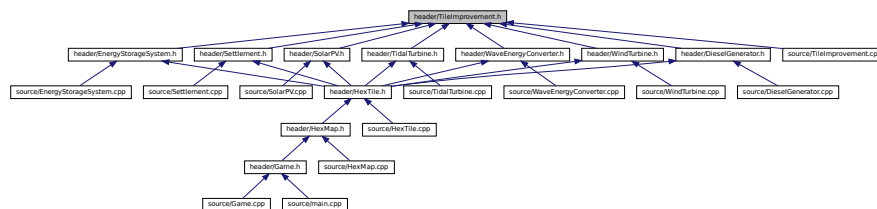
Classes

- class TileImprovement

  A base class for the tile improvement hierarchy.

Enumerations

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

  An enumeration of the different tile improvement types.

5.16.1 Detailed Description

Header file for the TileImprovement class.

5.16.2 Enumeration Type Documentation

5.16.2.1 TileImprovementType

enum TileImprovementType

An enumeration of the different tile improvement types.

Enumerator

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

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

5.17 header/WaveEnergyConverter.h File Reference

Header file for the WaveEnergyConverter class.

```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
#include "TileImprovement.h"
```
Include dependency graph for WaveEnergyConverter.h:
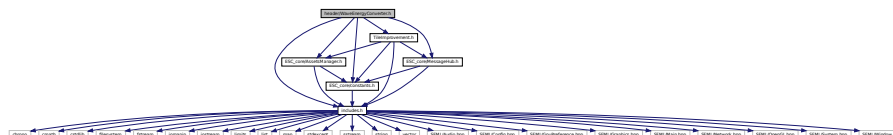
This graph shows which files directly or indirectly include this file:
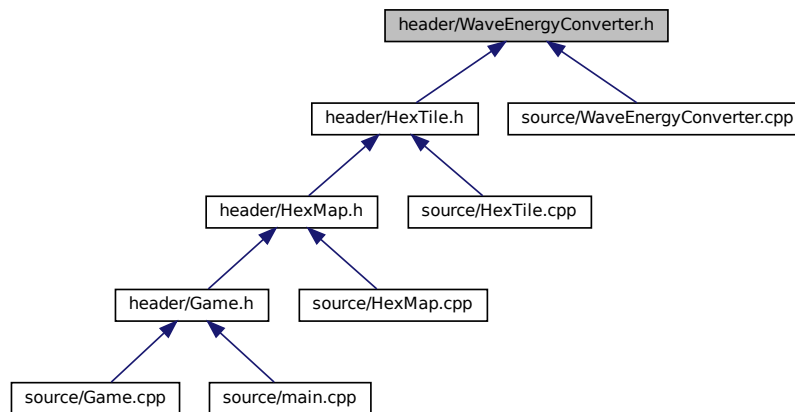
Classes

- class WaveEnergyConverter

  A settlement class (child class of TileImprovement).

5.17.1 Detailed Description

Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

Header file for the WindTurbine class.

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

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

Classes

- class WindTurbine

  A settlement class (child class of TileImprovement).

5.18.1 Detailed Description

Header file for the WindTurbine class.

5.19 source/ContextMenu.cpp File Reference

Implementation file for the ContextMenu class.

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

5.19.1 Detailed Description

Implementation file for the ContextMenu class.

A class which defines a context menu for the game.

5.20 source/DieselGenerator.cpp File Reference

Implementation file for the DieselGenerator class.

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

5.20.1 Detailed Description

Implementation file for the DieselGenerator class.

A base class for the tile improvement hierarchy.

5.21 source/EnergyStorageSystem.cpp File Reference

Implementation file for the EnergyStorageSystem class.

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

5.21.1 Detailed Description

Implementation file for the EnergyStorageSystem class.

A base class for the tile improvement hierarchy.

5.22 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:

5.22.1 Detailed Description

Implementation file for the AssetsManager class.

A class which manages visual and sound assets.

5.23 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:

5.23.1 Detailed Description

Implementation file for the MessageHub class.

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

5.24 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.

5.24.1 Detailed Description

Implementation file for various testing utilities.

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

5.24.2 Function Documentation

5.24.2.1 expectedErrorNotDetected()

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

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

Parameters

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__").

```
462 {
463     std::string error_str = "\n ERROR  failed to throw expected error prior to line ";
464     error_str += std::to_string(line);
```

```
465     error_str += " of ";
466     error_str += file;
467
468     #ifdef _WIN32
469         std::cout « error_str « std::endl;
470     #endif
471
472     throw std::runtime_error(error_str);
473     return;
474 }   /* expectedErrorNotDetected() */
```

5.24.2.2 printGold()

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

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

Parameters

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

```
114 {
115     std::cout « "\x1B[33m" « input_str « "\033[0m";
116     return;
117 }   /* printGold() */
```

5.24.2.3 printGreen()

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

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

Parameters

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

```
94 {
95     std::cout « "\x1B[32m" « input_str « "\033[0m";
96     return;
97 }   /* printGreen() */
```

5.24.2.4 printRed()

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

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

Parameters

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

```
134 {
135     std::cout « "\x1B[31m" « input_str « "\033[0m";
136     return;
137 }   /* printRed() */
```

5.24.2.5 testFloatEquals()

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

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

Parameters

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__").

```
168 {
169     if (fabs(x - y) <= FLOAT_TOLERANCE) {
170         return;
171     }
172
173     std::string error_str = "ERROR: testFloatEquals():\t in ";
174     error_str += file;
175     error_str += "\tline ";
176     error_str += std::to_string(line);
177     error_str += ":\t\n";
178     error_str += std::to_string(x);
179     error_str += " and ";
180     error_str += std::to_string(y);
181     error_str += " are not equal to within +/- ";
182     error_str += std::to_string(FLOAT_TOLERANCE);
183     error_str += "\n";
184
185     #ifdef _WIN32
186         std::cout « error_str « std::endl;
187     #endif
188
189     throw std::runtime_error(error_str);
190     return;
191 }   /* testFloatEquals() */
```

5.24.2.6 testGreaterThan()

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

Tests if x > y.

Parameters

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__").

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

5.24.2.7   testGreaterThanOrEqualTo()

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

Tests if x >= 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__").

```
272 {
273     if (x >= y) {
274         return;
275     }
276
277     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
278     error_str += file;
279     error_str += "\tline ";
280     error_str += std::to_string(line);
281     error_str += ":\t\n";
282     error_str += std::to_string(x);
283     error_str += " is not greater than or equal to ";
284     error_str += std::to_string(y);
285     error_str += "\n";
286
287     #ifdef _WIN32
288         std::cout « error_str « std::endl;
289     #endif
290
291     throw std::runtime_error(error_str);
```

```
292    return;
293 } /* testGreaterThanOrEqualTo() */
```

5.24.2.8 testLessThan()

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

Tests if x < y.

Parameters

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__").

```
323 {
324    if (x < y) {
325        return;
326    }
327
328    std::string error_str = "ERROR: testLessThan():\t in ";
329    error_str += file;
330    error_str += "\tline ";
331    error_str += std::to_string(line);
332    error_str += ":\t\n";
333    error_str += std::to_string(x);
334    error_str += " is not less than ";
335    error_str += std::to_string(y);
336    error_str += "\n";
337
338    #ifdef _WIN32
339        std::cout « error_str « std::endl;
340    #endif
341
342    throw std::runtime_error(error_str);
343    return;
344 } /* testLessThan() */
```

5.24.2.9 testLessThanOrEqualTo()

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

Tests if x <= 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__").

```
374 {
375     if (x <= y) {
376         return;
377     }
378
379     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
380     error_str += file;
381     error_str += "\tline ";
382     error_str += std::to_string(line);
383     error_str += ":\t\n";
384     error_str += std::to_string(x);
385     error_str += " is not less than or equal to ";
386     error_str += std::to_string(y);
387     error_str += "\n";
388
389     #ifdef _WIN32
390         std::cout « error_str « std::endl;
391     #endif
392
393     throw std::runtime_error(error_str);
394     return;
395 }   /* testLessThanOrEqualTo() */
```

5.24.2.10 testTruth()

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

Tests if the given statement is true.

Parameters

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__").

```
422 {
423     if (statement) {
424         return;
425     }
426
427     std::string error_str = "ERROR: testTruth():\t in ";
428     error_str += file;
429     error_str += "\tline ";
430     error_str += std::to_string(line);
431     error_str += ":\t\n";
432     error_str += "Given statement is not true";
433
434     #ifdef _WIN32
435         std::cout « error_str « std::endl;
436     #endif
437
438     throw std::runtime_error(error_str);
439     return;
440 }   /* testTruth() */
```

5.25 source/Game.cpp File Reference

Implementation file for the Game class.

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

5.25.1 Detailed Description

Implementation file for the Game class.

A class which defines a tile of a hex map.

5.26 source/HexMap.cpp File Reference

Implementation file for the HexMap class.

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

5.26.1 Detailed Description

Implementation file for the HexMap class.

A class which defines a hex map of hex tiles.

5.27 source/HexTile.cpp File Reference

Implementation file for the HexTile class.

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

5.27.1 Detailed Description

Implementation file for the HexTile class.

A class which defines a tile of a hex map.

5.28 source/main.cpp File Reference

Implementation file for main() for Road To Zero.

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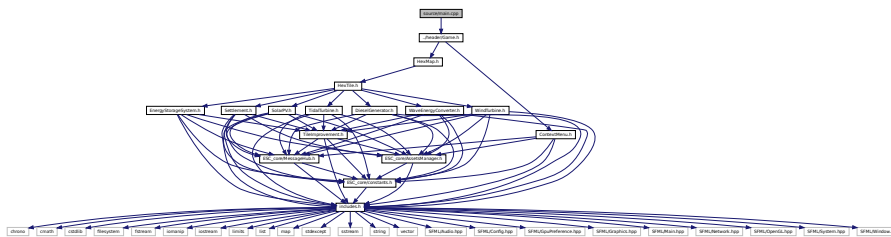
Functions

- 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)

5.28.1 Detailed Description

Implementation file for main() for Road To Zero.

5.28.2 Function Documentation

5.28.2.1 constructRenderWindow()

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

Helper function to construct render window.

Returns

Pointer to the render window.

```
294 {
295     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
296         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
297         "Road To Zero"
298     );
299
300     return render_window_ptr;
301 } /* constructRenderWindow() */
```

5.28.2.2 loadAssets()

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

Helper function to load game assets.

Parameters

assets_manager_ptr	Pointer to the assets manager.

```
66  {
67      //  1. load font assets
68      assets_manager_ptr->loadFont("assets/fonts/DroidSansMono.ttf", "DroidSansMono");
69      assets_manager_ptr->loadFont("assets/fonts/Glass_TTY_VT220.ttf", "Glass_TTY_VT220");
70
71
72      //  2. load tile sheets
73      assets_manager_ptr->loadTexture(
74          "assets/tile_sheets/pine_tree_64x64_1_CC-BY.png",
75          "pine_tree_64x64_1"
76      );
77
78      assets_manager_ptr->loadTexture(
79          "assets/tile_sheets/wheat_64x64_1_CC-BY.png",
80          "wheat_64x64_1"
81      );
82
83      assets_manager_ptr->loadTexture(
84          "assets/tile_sheets/mountain_64x64_1_CC-BY.png",
85          "mountain_64x64_1"
86      );
87
88      assets_manager_ptr->loadTexture(
89          "assets/tile_sheets/water_waves_64x64_1_CC-BY.png",
90          "water_waves_64x64_1"
91      );
92
93      assets_manager_ptr->loadTexture(
94          "assets/tile_sheets/water_shimmer_64x64_1_CC-BY.png",
95          "water_shimmer_64x64_1"
96      );
97
98      assets_manager_ptr->loadTexture(
99          "assets/tile_sheets/brick_house_64x64_1_CC-BY.png",
100          "brick_house_64x64_1"
101      );
102
103      assets_manager_ptr->loadTexture(
104          "assets/tile_sheets/magnifying_glass_64x64_1_CC-BY.png",
105          "magnifying_glass_64x64_1"
106      );
107
108      assets_manager_ptr->loadTexture(
109          "assets/tile_sheets/exp2_0_CC0.png",
110          "tile clear explosion"
111      );
112
113      assets_manager_ptr->loadTexture(
114          "assets/tile_sheets/emissions_8x8_1_CC-BY.png",
115          "emissions"
116      );
117
118      assets_manager_ptr->loadTexture(
119          "assets/tile_sheets/diesel_generator_64x64_2_CC-BY.png",
120          "diesel generator"
121      );
122
123      assets_manager_ptr->loadTexture(
124          "assets/tile_sheets/solar_PV_64x64_1_CC-BY.png",
125          "solar PV array"
126      );
127
128      assets_manager_ptr->loadTexture(
129          "assets/tile_sheets/wind_turbine_64x64_2_CC-BY.png",
130          "wind turbine"
131      );
132
133      assets_manager_ptr->loadTexture(
134          "assets/tile_sheets/energy_storage_system_64x64_1_CC-BY.png",
135          "energy storage system"
```

```
136     );
137
138     assets_manager_ptr->loadTexture(
139         "assets/tile_sheets/tidal_turbine_64x64_2_CC-BY.png",
140         "tidal turbine"
141     );
142
143     assets_manager_ptr->loadTexture(
144         "assets/tile_sheets/wave_energy_converter_64x64_2_CC-BY.png",
145         "wave energy converter"
146     );
147
148
149     //  3. load sounds
150     assets_manager_ptr->loadSound(
151         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
152         "coin ring"
153     );
154
155     assets_manager_ptr->loadSound(
156         "assets/audio/samples/mixkit-positive-notification-951_MixkitFree.ogg",
157         "positive notification"
158     );
159
160     assets_manager_ptr->loadSound(
161         "assets/audio/samples/mixkit-sci-fi-click-900_MixkitFree.ogg",
162         "sci-fi click"
163     );
164
165     assets_manager_ptr->loadSound(
166         "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932_MixkitFree.ogg",
167         "insufficient credits"
168     );
169
170     assets_manager_ptr->loadSound(
171         "assets/audio/samples/mixkit-data-scanner-2487_MixkitFree.ogg",
172         "resource assessment"
173     );
174
175     assets_manager_ptr->loadSound(
176         "assets/audio/samples/mixkit-interface-click-1126_MixkitFree.ogg",
177         "console string print"
178     );
179
180     assets_manager_ptr->loadSound(
181         "assets/audio/samples/mixkit-video-game-retro-click-237_MixkitFree.ogg",
182         "resource overlay toggle on"
183     );
184
185     assets_manager_ptr->loadSound(
186         "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED_MixkitFree.ogg",
187         "resource overlay toggle off"
188     );
189
190     assets_manager_ptr->loadSound(
191         "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703_MixkitFree.ogg",
192         "clear mountains tile"
193     );
194
195     assets_manager_ptr->loadSound(
196         "assets/audio/samples/mixkit-arcade-game-explosion-2759_MixkitFree.ogg",
197         "clear non-mountains tile"
198     );
199
200     assets_manager_ptr->loadSound(
201         "assets/audio/samples/mixkit-electronic-retro-block-hit-2185_MixkitFree.ogg",
202         "place improvement"
203     );
204
205     assets_manager_ptr->loadSound(
206         "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED_MixkitFree.ogg",
207         "build menu open"
208     );
209
210     assets_manager_ptr->loadSound(
211         "assets/audio/samples/mixkit-video-game-lock-2851_MixkitFree.ogg",
212         "build menu close"
213     );
214
215     assets_manager_ptr->loadSound(
216         "assets/audio/samples/mixkit-jump-into-the-water-1180_MixkitFree.ogg",
217         "splash"
218     );
219
220     assets_manager_ptr->loadSound(
221         "assets/audio/samples/505316__nuncaconoci__diesel_CC0.ogg",
222         "diesel running"
```

```
223        );
224
225        assets_manager_ptr->loadSound(
226            "assets/audio/samples/33460__pempi__320d_2_CC-BY.ogg",
227            "diesel start"
228        );
229
230        assets_manager_ptr->loadSound(
231            "assets/audio/samples/132724__andy_gardner__wind-turbine-blades_CC-BY.ogg",
232            "wind turbine running"
233        );
234
235        assets_manager_ptr->loadSound(
236            "assets/audio/samples/58416__darren1979__oceanwaves_CC-SAMPLING.ogg",
237            "ocean waves"
238        );
239
240        assets_manager_ptr->loadSound(
241            "assets/audio/samples/369927__mephisto_egmont__water-flowing-in-tubes_CC-BY.ogg",
242            "water flow"
243        );
244
245        assets_manager_ptr->loadSound(
246
        "assets/audio/samples/647663__jotraing__electric-train-motor-idle-loop-new-generation-rollingstock_CC0.ogg",
247            "energy storage system idle"
248        );
249
250        assets_manager_ptr->loadSound(
251            "assets/audio/samples/mixkit-epic-futuristic-movie-accent-2913_MixkitFree.ogg",
252            "game title screen"
253        );
254
255        assets_manager_ptr->loadSound(
256            "assets/audio/samples/mixkit-calm-park-with-people-and-children_MixkitFree.ogg",
257            "people and children"
258        );
259
260
261        //  4. load tracks
262        assets_manager_ptr->loadTrack(
263            "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
264            "Tree Star Moon - Dobranoc"
265        );
266
267        assets_manager_ptr->loadTrack(
268            "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
269            "Tree Star Moon - Lighthouse"
270        );
271
272        assets_manager_ptr->loadTrack(
273            "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
274            "Tree Star Moon - Sky Farm"
275        );
276
277        return;
278 }   /* loadAssets() */
```

5.28.2.3 main()

```
int main (
            int argc,
            char ** argv )
310 {
311      //  1. load assets
312      AssetsManager assets_manager;
313      loadAssets(&assets_manager);
314
315      //  2. construct render window
316      sf::RenderWindow* render_window_ptr = constructRenderWindow();
317
318      //  3. start game loop
319      bool quit_game = false;
320      assets_manager.playTrack();
321
322      while (not quit_game) {
323          Game game(render_window_ptr, &assets_manager);
324          quit_game = game.run();
325      }
```

```
326
327     //  4. clean up
328     render_window_ptr->close();
329     delete render_window_ptr;
330
331     return 0;
332 }   /* main() */
```

5.29 source/Settlement.cpp File Reference

Implementation file for the Settlement class.

```
#include "../header/Settlement.h"
```
Include dependency graph for Settlement.cpp:
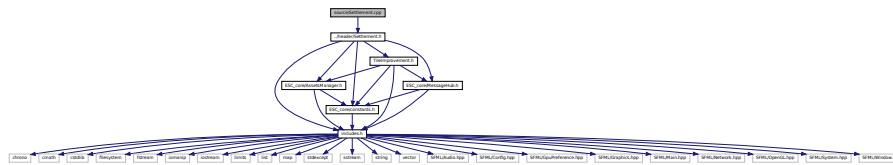
5.29.1 Detailed Description

Implementation file for the Settlement class.

A base class for the tile improvement hierarchy.

5.30 source/SolarPV.cpp File Reference

Implementation file for the SolarPV class.

```
#include "../header/SolarPV.h"
```
Include dependency graph for SolarPV.cpp:
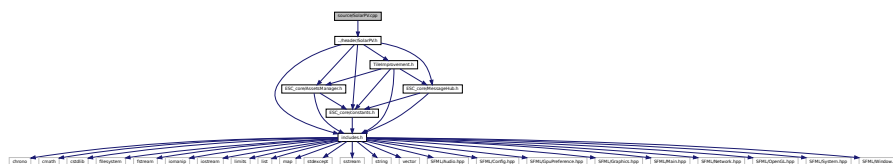
5.30.1 Detailed Description

Implementation file for the SolarPV class.

A base class for the tile improvement hierarchy.

5.31 source/TidalTurbine.cpp File Reference

Implementation file for the TidalTurbine class.

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

5.31.1 Detailed Description

Implementation file for the TidalTurbine class.

A base class for the tile improvement hierarchy.

5.32 source/TileImprovement.cpp File Reference

Implementation file for the TileImprovement class.

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

5.32.1 Detailed Description

Implementation file for the TileImprovement class.

A base class for the tile improvement hierarchy.

5.33 source/WaveEnergyConverter.cpp File Reference

Implementation file for the WaveEnergyConverter class.

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

5.33.1 Detailed Description

Implementation file for the WaveEnergyConverter class.

A base class for the tile improvement hierarchy.

5.34 source/WindTurbine.cpp File Reference

Implementation file for the WindTurbine class.

```
#include "../header/WindTurbine.h"
```
Include dependency graph for WindTurbine.cpp:
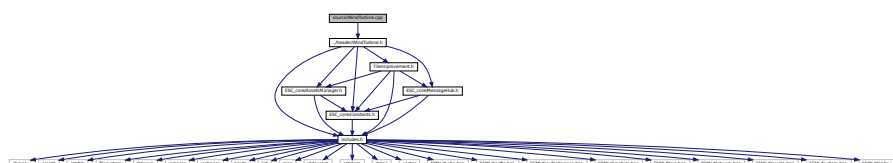
5.34.1 Detailed Description

Implementation file for the WindTurbine 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/. 193

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 192

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 39, 46, 91, 140, 147, 152, 158, 167, 172

Index

WindTurbine, 170
 __handleKeyPressEvents, 173
 __handleMouseButtonEvents, 173
 __setUpTileImprovementSpriteAnimated, 173
 ~WindTurbine, 172
 draw, 174
 processEvent, 174
 processMessage, 175
 WindTurbine, 172

year
 Game, 64