Road To Zero - The Microgrid Management Game

Generated by Doxygen 1.9.1

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:

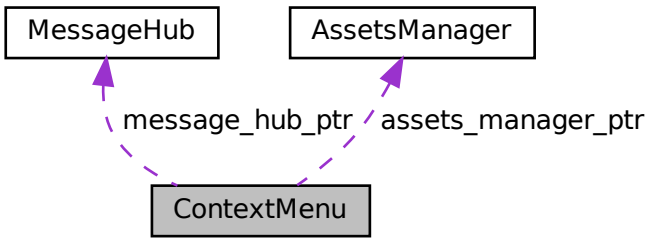
- header/ESC_core/AssetsManager.h
- source/ESC_core/AssetsManager.cpp

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:

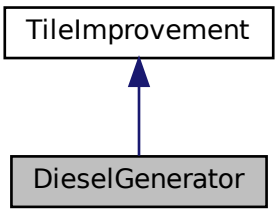
- header/ContextMenu.h
- source/ContextMenu.cpp

4.3 DieselGenerator Class Reference

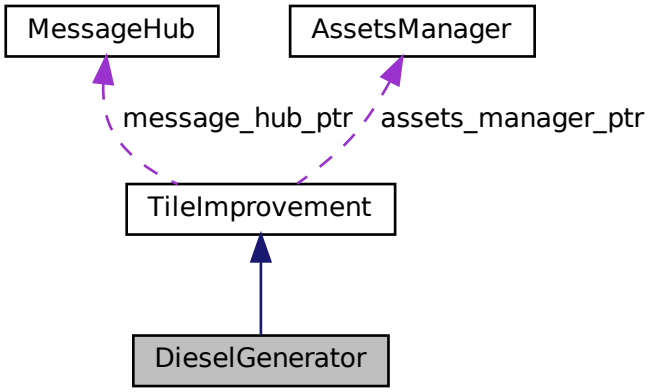
A settlement class (child class of TileImprovement).

```
#include <DieselGenerator.h>
```

Inheritance diagram for DieselGenerator:



Collaboration diagram for DieselGenerator:



Public Member Functions

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

  Constructor for the DieselGenerator class.
- std::string getTileOptionsSubstring (void)

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

  Method to set the is selected attribute.
- void advanceTurn (void)

  Method to handle turn advance.
- void processEvent (void)

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

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

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

  Destructor for the DieselGenerator class.

Public Attributes

- int capacity_kW

  The rated production capacity [kW] of the diesel generator.
- int production_MWh

  The current production [MWh] of the diesel generator.
- int max_production_MWh

  The maximum production [MWh] for this turn.
- double smoke_da

  The per frame delta in smoke particle alpha value.

- double smoke_dx

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

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

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

  A list of smoke sprite (for exhaust animation).
- int fuel_cost

  The fuel costs for this turn.
- int emissions_tonnes_CO2e

  The emissions for this turn.

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

  Helper method to draw production menu assets.
- void __upgrade (void)

  Helper method to upgrade the diesel generator.
- void __computeProductionCosts (void)

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

  Helper method to trigger an equipment breakdown.
- void __handleKeyPressEvents (void)

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

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

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.3.1 Detailed Description

A settlement class (child class of TileImprovement).

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DieselGenerator()

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

Constructor for the DieselGenerator class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

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

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

4.3.2.2 ~DieselGenerator()

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

Destructor for the DieselGenerator class.

```
861 {
862     std::cout « "DieselGenerator at " « this « " destroyed" « std::endl;
863
864     return;
865 }   /* ~DieselGenerator() */
```

4.3.3 Member Function Documentation

4.3.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.3.3.2 __computeProductionCosts()

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

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

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

4.3.3.3 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.3.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

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

4.3.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

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

4.3.3.6 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

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

4.3.3.7 __setUpTileImprovementSpriteAnimated()

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

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

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

4.3.3.8 __upgrade()

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

Helper method to upgrade the diesel generator.

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

4.3.3.9 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
616 {
617     //  1. send improvement state message
618     this->__sendImprovementStateMessage();
619 
620     //  2. handle start/stop
621     if ((not this->is_running) and (this->production_MWh > 0)) {
622         this->is_running = true;
623         this->assets_manager_ptr->getSound("diesel start")->play();
624     }
625 
626     else if (this->is_running and (this->production_MWh <= 0)) {
627         this->is_running = false;
628         this->tile_improvement_sprite_animated[1].setScale(sf::Vector2f(1, 1));
629     }
630 
631     //  3. handle equipment health
632     if (this->is_running) {
633         this->health--;
634 
635         if (this->health <= 0) {
636             this->__breakdown();
637         }
638     }
639 
640     //  4. close menus
641     if (this->production_menu_open) {
```

```
642         this->__closeProductionMenu();
643     }
644
645     if (this->upgrade_menu_open) {
646         this->__closeUpgradeMenu();
647     }
648
649     return;
650 }   /* advanceTurn() */
```

4.3.3.10 draw()

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

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

Reimplemented from TileImprovement.

```
714 {
715     //  1. if just built, call base method and return
716     if (this->just_built) {
717         TileImprovement :: draw();
718
719         return;
720     }
721
722     //  2. handle upgrade effects
723     if (this->just_upgraded) {
724         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
725             this->tile_improvement_sprite_animated[i].setColor(
726                 sf::Color(
727                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
728                     255,
729                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
730                     255
731                 )
732             );
733
734             this->tile_improvement_sprite_animated[i].setScale(
735                 sf::Vector2f(
736                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
737                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
738                 )
739             );
740         }
741
742         this->upgrade_frame++;
743     }
744
745     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
746         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
747             this->tile_improvement_sprite_animated[i].setColor(
748                 sf::Color(255,255,255,255)
749             );
750
751             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
752         }
753
754         this->just_upgraded = false;
755         this->upgrade_frame = 0;
756     }
757
758
759     //  3. draw first element of animated sprite
760     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
761
762
763     //  4. draw second element of animated sprite
764     double move_x = 0;
765     double move_y = 0;
766
767     if (this->is_running) {
768         this->tile_improvement_sprite_animated[1].setScale(
769             sf::Vector2f(
770                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2),
771                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2)
772             )
```

```
773         );
774
775         move_x = 1 * ((double)rand() / RAND_MAX) - 0.5;
776         move_y = 1 * ((double)rand() / RAND_MAX) - 0.5;
777
778         this->tile_improvement_sprite_animated[1].move(move_x, move_y);
779     }
780
781     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
782
783     if (this->is_running) {
784         this->tile_improvement_sprite_animated[1].move(-1 * move_x, -1 * move_y);
785     }
786
787
788     //  5. draw smoke effects
789     if (this->is_running) {
790         if ((double)rand() / RAND_MAX < smoke_prob) {
791             this->smoke_sprite_list.push_back(
792                 sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
793             );
794
795             this->smoke_sprite_list.back().setOrigin(
796                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
797                 this->smoke_sprite_list.back().getLocalBounds().height / 2
798             );
799
800             this->smoke_sprite_list.back().setPosition(
801                 this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
802                 this->position_y - 33
803             );
804         }
805     }
806
807     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
808
809     double alpha = 255;
810
811     while (iter != this->smoke_sprite_list.end()) {
812         this->render_window_ptr->draw(*iter);
813
814         alpha = (*iter).getColor().a;
815
816         alpha -= this->smoke_da;
817
818         if (alpha <= 0) {
819             iter = this->smoke_sprite_list.erase(iter);
820             continue;
821         }
822
823         (*iter).setColor(sf::Color(255, 255, 255, alpha));
824
825         (*iter).move(
826             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
827             this->smoke_dy
828         );
829
830         (*iter).rotate(((double)rand() / RAND_MAX));
831
832         iter++;
833     }
834
835
836     //  6. draw production menu
837     if (this->production_menu_open) {
838         this->render_window_ptr->draw(this->production_menu_backing);
839         this->render_window_ptr->draw(this->production_menu_backing_text);
840
841         this->__drawProductionMenu();
842     }
843
844     this->frame++;
845     return;
846 }   /* draw() */
```

4.3.3.11 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

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

4.3.3.12 processEvent()

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

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

Reimplemented from TileImprovement.

```
665 {
666     TileImprovement :: processEvent();
667 
668     if (this->event_ptr->type == sf::Event::KeyPressed) {
669         this->__handleKeyPressEvents();
670     }
671 
672     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
673         this->__handleMouseButtonEvents();
674     }
675 
676     return;
677 }   /* processEvent() */
```

4.3.3.13 processMessage()

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

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

Reimplemented from TileImprovement.

```
692 {
693     TileImprovement :: processMessage();
694
695     //...
696
697     return;
698 }   /* processMessage() */
```

4.3.3.14 setIsSelected()

```
void DieselGenerator::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
593 {
594     TileImprovement :: setIsSelected(is_selected);
595
596     if (this->is_running and this->is_selected) {
597         this->assets_manager_ptr->getSound("diesel running")->play();
598     }
599
600     return;
601 }   /* setIsSelected() */
```

4.3.4 Member Data Documentation

4.3.4.1 capacity_kW

```
int DieselGenerator::capacity_kW
```

The rated production capacity [kW] of the diesel generator.

4.3.4.2 emissions_tonnes_CO2e

```
int DieselGenerator::emissions_tonnes_CO2e
```

The emissions for this turn.

4.3.4.3 fuel_cost

```
int DieselGenerator::fuel_cost
```

The fuel costs for this turn.

4.3.4.4 max_production_MWh

```
int DieselGenerator::max_production_MWh
```

The maximum production [MWh] for this turn.

4.3.4.5 production_MWh

```
int DieselGenerator::production_MWh
```

The current production [MWh] of the diesel generator.

4.3.4.6 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.7 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.8 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.9 smoke_prob

```
double DieselGenerator::smoke_prob
```

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

4.3.4.10 smoke_sprite_list

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

A list of smoke sprite (for exhaust animation).

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

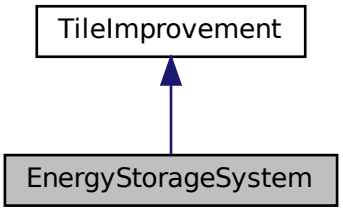
- header/DieselGenerator.h
- source/DieselGenerator.cpp

4.4 EnergyStorageSystem Class Reference

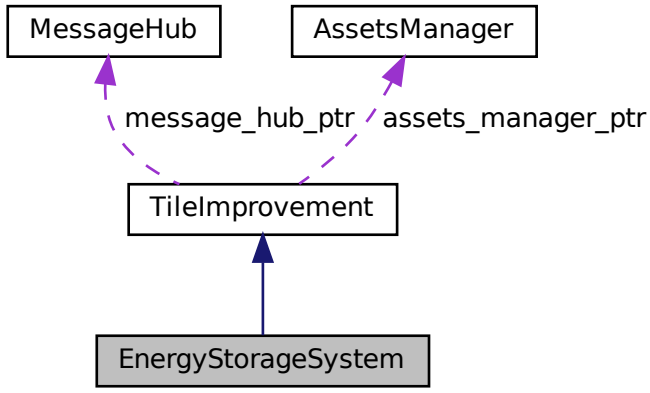
A settlement class (child class of TileImprovement).

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

Inheritance diagram for EnergyStorageSystem:



Collaboration diagram for EnergyStorageSystem:



Public Member Functions

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

  Constructor for the EnergyStorageSystem class.
- void setIsSelected (bool)

  Method to set the is selected attribute.
- std::string getTileOptionsSubstring (void)

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

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

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

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

  Destructor for the EnergyStorageSystem class.

Public Attributes

- int capacity_MWh

  The rated energy capacity [MWh] of the energy storage system.
- int charge_MWh

  The charge [MWh] in the energy storage system.

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

  Helper method to set up and position production menu assets (drawable).
- void __upgrade (void)

  Helper method to upgrade the diesel generator.
- void __handleKeyPressEvents (void)

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.4.1 Detailed Description

A settlement class (child class of TileImprovement).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 EnergyStorageSystem()

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

Constructor for the EnergyStorageSystem class.

Ref: Wikipedia [2023]

Parameters

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.

```
291   :
292 TileImprovement(
```

```
293     position_x,
294     position_y,
295     event_ptr,
296     render_window_ptr,
297     assets_manager_ptr,
298     message_hub_ptr
299 )
300 {
301     //  1. set attributes
302 
303     //  1.1. private
304     //...
305 
306     //  1.2. public
307     this->tile_improvement_type = TileImprovementType :: ENERGY_STORAGE_SYSTEM;
308 
309     this->is_running = false;
310 
311     this->health = 100;
312 
313     this->capacity_MWh = 1;
314     this->upgrade_level = 1;
315 
316     this->charge_MWh = 0;
317 
318     this->tile_improvement_string = "ENERGY STORAGE";
319 
320     this->__setUpTileImprovementSpriteStatic();
321     this->__setUpProductionMenu();
322 
323     std::cout « "EnergyStorageSystem constructed at " « this « std::endl;
324 
325     return;
326 }   /* EnergyStorageSystem() */
```

4.4.2.2 ~EnergyStorageSystem()

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

Destructor for the EnergyStorageSystem class.

```
504 {
505     std::cout « "EnergyStorageSystem at " « this « " destroyed" « std::endl;
506 
507     return;
508 }   /* ~EnergyStorageSystem() */
```

4.4.3 Member Function Documentation

4.4.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
179 {
180     if (this->just_built) {
181         return;
182     }
183 
184     switch (this->event_ptr->key.code) {
185         case (sf::Keyboard::U): {
186             if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
187                 this->__upgrade();
188             }
```

```
189
190             break;
191         }
192
193
194         default: {
195             // do nothing!
196
197             break;
198         }
199     }
200
201     return;
202 }   /* __handleKeyPressEvents() */
```

4.4.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
217 {
218     if (this->just_built) {
219         return;
220     }
221
222     switch (this->event_ptr->mouseButton.button) {
223         case (sf::Mouse::Left): {
224             //...
225
226             break;
227         }
228
229
230         case (sf::Mouse::Right): {
231             //...
232
233             break;
234         }
235
236
237         default: {
238             // do nothing!
239
240             break;
241         }
242     }
243
244     return;
245 }   /* __handleMouseButtonEvents() */
```

4.4.3.3 __setUpProductionMenu()

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

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

```
103 {
104     //  1. modify production menu text
105     this->production_menu_backing_text.setString("**** DISCHARGE MENU ****");
106     this->production_menu_backing_text.setFont(
107         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
108     );
109     this->production_menu_backing_text.setCharacterSize(16);
110     this->production_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
111     this->production_menu_backing_text.setOrigin(
112         this->production_menu_backing_text.getLocalBounds().width / 2, 0
113     );
114     this->production_menu_backing_text.setPosition(400, 400 - 128 + 4);
115
116     return;
117 }   /* __setUpProductionMenu() */
```

4.4.3.4 __setUpTileImprovementSpriteStatic()

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

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

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

4.4.3.5 __upgrade()

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

Helper method to upgrade the diesel generator.

```
132 {
133     /*
134     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
135
136     if (this->credits < upgrade_cost) {
137         std::cout « "Cannot upgrade diesel generator: insufficient credits (need "
138             « upgrade_cost « " K)" « std::endl;
139
140         this->__sendInsufficientCreditsMessage();
141         return;
142     }
143
144     this->is_running = false;
145
146     this->health = 100;
147
148     this->capacity_kW += 100;
149     this->upgrade_level++;
150
151     this->production_MWh = 0;
152     this->max_production_MWh += 72;
153
154     this->just_upgraded = true;
155
156     this->assets_manager_ptr->getSound("upgrade")->play();
157
158     this->__sendCreditsSpentMessage(upgrade_cost);
159     this->__sendTileStateRequest();
160     this->__sendGameStateRequest();
161     */
162
163     return;
164 }   /* __upgrade() */
```

4.4.3.6 draw()

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

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

Reimplemented from TileImprovement.

```
466 {
467     //  1. if just built, call base method and return
468     if (this->just_built) {
469         TileImprovement :: draw();
470 
471         return;
472     }
473 
474 
475     //  2. draw static sprite
476     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
477 
478 
479     //  3. draw production menu
480     if (this->production_menu_open) {
481         this->render_window_ptr->draw(this->production_menu_backing);
482         this->render_window_ptr->draw(this->production_menu_backing_text);
483 
484         //...
485     }
486 
487     this->frame++;
488     return;
489 }   /* draw() */
```

4.4.3.7 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
368 {
369     int upgrade_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
370 
371     //                  32 char x 17 line console "--------------------------------\n";
372     std::string options_substring                = "CAPACITY:  ";
373     options_substring                           += std::to_string(this->capacity_MWh);
374     options_substring                           += " MWh (level ";
375     options_substring                           += std::to_string(this->upgrade_level);
376     options_substring                           += ")\n";
377 
378     options_substring                           += "CHARGE:    ";
379     options_substring                           += std::to_string(this->charge_MWh);
380     options_substring                           += " MWh\n";
381 
382     options_substring                           += "HEALTH:    ";
383     options_substring                           += std::to_string(this->health);
384     options_substring                           += "/100\n";
385 
386     options_substring                           += "                                \n";
387     options_substring                           += "**** ENERGY STORAGE OPTIONS ****\n";
388     options_substring                           += "                                \n";
389     options_substring                           += "     [E]:  OPEN DISCHARGE MENU  \n";
390 
391     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
392         options_substring                            += "     [U]:  UPGRADE (";
```

```
393         options_substring                           += std::to_string(upgrade_cost);
394         options_substring                           +=" K)\n";
395     }
396
397     options_substring                           += "HOLD [P]:  SCRAP (";
398     options_substring                           += std::to_string(SCRAP_COST);
399     options_substring                           += " K)";
400
401     return options_substring;
402 }   /* getTileOptionsSubstring() */
```

4.4.3.8 processEvent()

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

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

Reimplemented from TileImprovement.

```
417 {
418     TileImprovement :: processEvent();
419
420     if (this->event_ptr->type == sf::Event::KeyPressed) {
421         this->__handleKeyPressEvents();
422     }
423
424     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
425         this->__handleMouseButtonEvents();
426     }
427
428     return;
429 }   /* processEvent() */
```

4.4.3.9 processMessage()

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

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

Reimplemented from TileImprovement.

```
444 {
445     TileImprovement :: processMessage();
446
447     //...
448
449     return;
450 }   /* processMessage() */
```

4.4.3.10 setIsSelected()

```
void EnergyStorageSystem::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
343 {
344     TileImprovement :: setIsSelected(is_selected);
345 
346     if (this->is_selected) {
347         this->assets_manager_ptr->getSound("energy storage system")->play();
348     }
349 
350     return;
351 }   /* setIsSelected() */
```

4.4.4 Member Data Documentation

4.4.4.1 capacity_MWh

```
int EnergyStorageSystem::capacity_MWh
```

The rated energy capacity [MWh] of the energy storage system.

4.4.4.2 charge_MWh

```
int EnergyStorageSystem::charge_MWh
```

The charge [MWh] in the energy storage system.

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

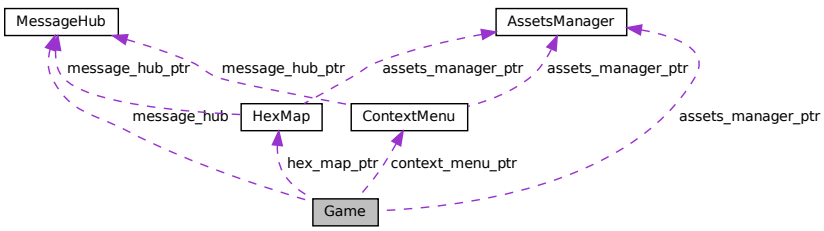
- header/EnergyStorageSystem.h
- source/EnergyStorageSystem.cpp

4.5 Game Class Reference

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

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

Collaboration diagram for Game:



Public Member Functions

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

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

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

  Destructor for the Game class.

Public Attributes

- GamePhase game_phase

  The current phase of the game.
- bool quit_game

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

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

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

  Boolean indicating whether or not to check terminating conditions.
- bool message_deadlock

  A boolean indicating whether a message deadlock has been detected.
- unsigned long long int frame

  The current frame of the game.
- double time_since_start_s

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

  Current game year.
- int month

  Current game month.
- int population

  Current population.
- int credits

  Current balance of credits.
- int demand_MWh

  Current energy demand [MWh].
- int demand_remaining_MWh

  The current remaining energy demand [MWh].
- int cumulative_emissions_tonnes

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

  A frame counter for detecting message deadlock.
- int turn = 0

  The current game turn.
- std::vector< double > demand_vec_MWh

  A vector of daily demands [MWh] for the current month.
- sf::Clock clock

  The game clock.
- sf::Event event

The game events class.

- MessageHub message_hub

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

- HexMap * hex_map_ptr

  Pointer to the hex map (defines game world).

- ContextMenu * context_menu_ptr

  Pointer to the context menu.

Private Member Functions

- void __toggleFrameClockOverlay (void)

  Helper method to toggle frame clock overlay.

- void __checkTerminatingConditions (void)

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

- void __advanceTurn (void)

  Helper method to advance turn.

- void __computeCurrentDemand (void)

  Helper method to compute current energy demand.

- void __handleKeyPressEvents (void)

  Helper method to handle key press events.

- void __handleMouseButtonEvents (void)

  Helper method to handle mouse button events.

- void __handleImprovementStateMessage (Message)

  Helper method to handle improvement state messages.

- void __processEvent (void)

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

- void __processMessage (void)

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

- void __sendGameStateMessage (void)

  Helper method to format and send a game state message.

- void __sendTurnAdvanceMessage (void)

  Helper method to format and send a turn advance message.

- void __sendCreditsEarnedMessage (void)

  Helper method to format and send a credits earned message.

- void __insufficientCreditsAlarm (void)

  Helper method to sound and display and insufficient credits alarm.

- void __drawFrameClockOverlay (void)

  Helper method to draw frame clock overlay.

- void __drawHUD (void)

  Helper method to heads-up display (HUD).

- void __draw (void)

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

Private Attributes

- sf::RenderWindow * render_window_ptr

  A pointer to the render window.

- AssetsManager * assets_manager_ptr

  A pointer to the assets manager.

4.5.1 Detailed Description

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

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Game()

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

Constructor for the Game class.

```
920 {
921     //  1. set attributes
922 
923     //  1.1. private
924     this->render_window_ptr = render_window_ptr;
925 
926     this->assets_manager_ptr = assets_manager_ptr;
927 
928     //  1.2. public
929     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
930 
931     this->quit_game = false;
932     this->game_loop_broken = false;
933     this->show_frame_clock_overlay = false;
934     this->check_terminating_conditions = false;
935 
936     this->frame = 0;
937     this->time_since_start_s = 0;
938 
939     this->message_deadlock = false;
940     this->message_deadlock_frame = 0;
941 
942     double seconds_since_epoch = time(NULL);
943     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
944 
945     this->year = 1970 + (int)years_since_epoch;
946     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
947     while (this->month > 12) {
948         this->month -= 12;
949     }
950 
951     this->population = 0;
952     this->credits = STARTING_CREDITS;
953     this->demand_MWh = 0;
954     this->demand_remaining_MWh = 0;
955     this->cumulative_emissions_tonnes = 0;
956 
957     this->demand_vec_MWh.resize(30, 0);
958 
959     this->hex_map_ptr = new HexMap(
960         6,
961         &(this->event),
962         this->render_window_ptr,
963         this->assets_manager_ptr,
964         &(this->message_hub)
965     );
966 
967     this->context_menu_ptr = new ContextMenu(
968         &(this->event),
969         this->render_window_ptr,
970         this->assets_manager_ptr,
971         &(this->message_hub)
972     );
973 
974     //  2. add message channel(s)
975     this->message_hub.addChannel(GAME_CHANNEL);
976     this->message_hub.addChannel(GAME_STATE_CHANNEL);
```

```
977
978     std::cout « "Game constructed at " « this « std::endl;
979
980     return;
981 }   /* Game() */
```

4.5.2.2 ~Game()

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

Destructor for the Game class.

```
1085 {
1086     //  1. clean up attributes
1087     delete this->hex_map_ptr;
1088     delete this->context_menu_ptr;
1089
1090     std::cout « "Game at " « this « " destroyed" « std::endl;
1091
1092     return;
1093 }   /* ~Game() */
```

4.5.3 Member Function Documentation

4.5.3.1 __advanceTurn()

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

Helper method to advance turn.

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

4.5.3.2 __checkTerminatingConditions()

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

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

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

4.5.3.3 __computeCurrentDemand()

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

Helper method to compute current energy demand.

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

4.5.3.4 __draw()

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

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

```
887 {
888     this->__drawHUD();
889
890     if (this->show_frame_clock_overlay) {
891         this->__drawFrameClockOverlay();
892     }
893
894     return;
895 }   /* draw() */
```

4.5.3.5 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
713 {
714     std::string frame_clock_string = "FRAME: ";
715     frame_clock_string += std::to_string(this->frame);
716     frame_clock_string += "\nTIME SINCE START [s]: ";
717     frame_clock_string += std::to_string(this->time_since_start_s);
718 
719     sf::Text frame_clock_text(
720         frame_clock_string,
721         *(this->assets_manager_ptr->getFont("DroidSansMono")),
722         16
723     );
724 
725     sf::RectangleShape frame_clock_backing(
726         sf::Vector2f(
727             1.02 * frame_clock_text.getLocalBounds().width,
728             1.20 * frame_clock_text.getLocalBounds().height
729         )
730     );
731     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
732 
733     this->render_window_ptr->draw(frame_clock_backing);
734     this->render_window_ptr->draw(frame_clock_text);
735 
736     return;
737 }   /* __drawFrameClockOverlay() */
```

4.5.3.6 __drawHUD()

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

Helper method to heads-up display (HUD).

```
752 {
753     //  1. first line (top)
754     std::string HUD_string = "YEAR: ";
755     HUD_string += std::to_string(this->year);
756 
757     HUD_string += "    MONTH: ";
758     HUD_string += std::to_string(this->month);
759 
760     HUD_string += "    POPULATION: ";
761     HUD_string += std::to_string(this->population);
762 
763     HUD_string += "    CREDITS: ";
764     HUD_string += std::to_string(this->credits);
765     HUD_string += " K";
766 
767     HUD_string += "    CURRENT DEMAND: ";
768     HUD_string += std::to_string(this->demand_MWh);
769     HUD_string += " MWh";
770 
771     sf::Text HUD_text(
772         HUD_string,
773         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
774         16
775     );
776 
777     HUD_text.setPosition(
778         (800 - HUD_text.getLocalBounds().width) / 2,
779         8
780     );
781 
782     HUD_text.setFillColor(MONOCHROME_TEXT_GREEN);
783 
784     this->render_window_ptr->draw(HUD_text);
785 
786 
787     //  2. second line (top)
788     HUD_string = "CUMULATIVE EMISSIONS: ";
```

```
789     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
790     HUD_string += " tonnes (CO2e)";
791 
792     HUD_string += "    LIFETIME LIMIT: ";
793     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
794     HUD_string += " tonnes (CO2e)";
795 
796     HUD_text.setString(HUD_string);
797 
798     HUD_text.setPosition(
799         (800 - HUD_text.getLocalBounds().width) / 2,
800         35
801     );
802 
803     this->render_window_ptr->draw(HUD_text);
804 
805 
806     //  3. third line (bottom)
807     HUD_string = "GAME PHASE: ";
808 
809     switch (this->game_phase) {
810         case (GamePhase :: BUILD_SETTLEMENT): {
811             HUD_string += "BUILD SETTLEMENT";
812 
813             break;
814         }
815 
816 
817         case (GamePhase :: SYSTEM_MANAGEMENT): {
818             HUD_string += "SYSTEM MANAGEMENT";
819 
820             break;
821         }
822 
823 
824         case (GamePhase :: LOSS_EMISSIONS): {
825             HUD_string += "LOSS (EMISSIONS)";
826 
827             break;
828         }
829 
830 
831         case (GamePhase :: LOSS_DEMAND): {
832             HUD_string += "LOSS (DEMAND)";
833 
834             break;
835         }
836 
837 
838         case (GamePhase :: LOSS_CREDITS): {
839             HUD_string += "LOSS (CREDITS)";
840 
841             break;
842         }
843 
844 
845         case (GamePhase :: VICTORY): {
846             HUD_string += "VICTORY";
847 
848             break;
849         }
850 
851 
852         default: {
853             HUD_string += "???";
854 
855             break;
856         }
857     }
858 
859     HUD_string += "    TURN: ";
860     HUD_string += std::to_string(this->turn);
861 
862     HUD_text.setString(HUD_string);
863 
864     HUD_text.setPosition(
865         (800 - HUD_text.getLocalBounds().width) / 2,
866         GAME_HEIGHT - 35
867     );
868 
869     this->render_window_ptr->draw(HUD_text);
870 
871     return;
872 }   /* __drawHUD() */
```

4.5.3.7 __handleImprovementStateMessage()

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

Helper method to handle improvement state messages.

```
280 {
281     //  1. unpack message and update game attributes
282     if (improvement_state_message.int_payload.count("dispatch_MWh") > 0) {
283         this->demand_remaining_MWh -= improvement_state_message.int_payload["dispatch_MWh"];
284 
285         this->credits +=
286             round(CREDITS_PER_MWH_SERVED * improvement_state_message.int_payload["dispatch_MWh"]);
287 
288         this->__sendCreditsEarnedMessage();
289     }
290 
291     if (improvement_state_message.int_payload.count("fuel_cost") > 0) {
292         this->credits -= improvement_state_message.int_payload["fuel_cost"];
293     }
294 
295     if (improvement_state_message.int_payload.count("operation_maintenance_cost") > 0) {
296         this->credits -=
297             improvement_state_message.int_payload["operation_maintenance_cost"];
298     }
299 
300     if (improvement_state_message.int_payload.count("emissions_tonnes_CO2e") > 0) {
301         this->cumulative_emissions_tonnes +=
302             improvement_state_message.int_payload["emissions_tonnes_CO2e"];
303     }
304 
305     return;
306 }   /* __handleImprovementStateMessage() */
```

4.5.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

4.5.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
241 {
242     switch (this->event.mouseButton.button) {
243         case (sf::Mouse::Left): {
244             //...
245 
246             break;
247         }
248 
249 
250         case (sf::Mouse::Right): {
251             //...
252 
253             break;
254         }
255 
256 
257         default: {
258             // do nothing!
259 
260             break;
261         }
262     }
263 
264     return;
265 }   /* __handleMouseButtonEvents() */
```

4.5.3.10 __insufficientCreditsAlarm()

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

Helper method to sound and display and insufficient credits alarm.

```
606 {
607     //  1. sound buzzer
608     this->assets_manager_ptr->getSound("insufficient credits")->play();
609 
610     //  2. construct alarm text and backing rectangle
611     sf::Text insufficient_credits_text(
612         "INSUFFICIENT CREDITS",
613         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
614         32
615     );
616 
617     insufficient_credits_text.setOrigin(
618         insufficient_credits_text.getLocalBounds().width / 2,
619         insufficient_credits_text.getLocalBounds().height / 2
620     );
621 
622     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
623 
624     sf::RectangleShape backing_rectangle(
625         sf::Vector2f(
626             1.1 * insufficient_credits_text.getLocalBounds().width,
627             1.5 * insufficient_credits_text.getLocalBounds().height
628         )
629     );
630 
631     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
632 
633     backing_rectangle.setOrigin(
634         backing_rectangle.getLocalBounds().width / 2,
635         backing_rectangle.getLocalBounds().height / 2
636     );
637 
638     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
639 
640     //  3. display loop (blocking ~3 seconds)
641     bool red_flag = true;
642     int alarm_frame = 0;
```

```
643     double time_since_alarm_s = 0;
644
645     sf::Clock alarm_clock;
646
647     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
648
649
650         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
651
652         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
653             while (this->render_window_ptr->pollEvent(this->event)) {
654                 // do nothing!
655             }
656
657             this->render_window_ptr->clear();
658
659             this->hex_map_ptr->draw();
660             this->context_menu_ptr->draw();
661             this->__draw();
662
663             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
664                 if (red_flag) {
665                     red_flag = false;
666                 }
667
668                 else {
669                     red_flag = true;
670                 }
671             }
672
673             if (red_flag) {
674                 insufficient_credits_text.setFillColor(MONOCHROME_TEXT_RED);
675             }
676
677             else {
678                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
679             }
680
681             this->render_window_ptr->draw(backing_rectangle);
682             this->render_window_ptr->draw(insufficient_credits_text);
683
684             this->render_window_ptr->display();
685
686             alarm_frame++;
687             this->frame++;
688         }
689
690         // check track status, move to next if stopped
691         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
692             this->assets_manager_ptr->nextTrack();
693             this->assets_manager_ptr->playTrack();
694         }
695     }
696
697     return;
698 }   /* __insufficientCreditsAlarm( */
```

4.5.3.11 __processEvent()

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

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

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

4.5.3.12 __processMessage()

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

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

```
486 {
487     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
488         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
489 
490         if (game_channel_message.subject == "quit game") {
491             this->quit_game = true;
492             this->game_loop_broken = true;
493 
494             std::cout « "Quit game message received by " « this « std::endl;
495             this->message_hub.popMessage(GAME_CHANNEL);
496         }
497 
498         if (game_channel_message.subject == "restart game") {
499             this->game_loop_broken = true;
500 
501             std::cout « "Restart game message received by " « this « std::endl;
502             this->message_hub.popMessage(GAME_CHANNEL);
503         }
504 
505         if (game_channel_message.subject == "state request") {
506             std::cout « "Game state request message received by " « this « std::endl;
507 
508             this->__sendGameStateMessage();
509             this->message_hub.popMessage(GAME_CHANNEL);
510         }
511 
512         if (game_channel_message.subject == "credits spent") {
513             this->credits -= game_channel_message.int_payload["credits spent"];
514 
515             std::cout « "Credits spent message (" «
516                 game_channel_message.int_payload["credits spent"] « ") received by "
517                 « this « std::endl;
518 
519             std::cout « "Current credits (Game): " « this->credits « " K" «
520                 std::endl;
521 
522             this->message_hub.popMessage(GAME_CHANNEL);
523         }
524 
525         if (game_channel_message.subject == "insufficient credits") {
526             std::cout « "Insufficient credits message received by " « this «
527                 std::endl;
528 
529             this->__insufficientCreditsAlarm();
530 
531             this->message_hub.popMessage(GAME_CHANNEL);
532         }
533 
534         if (game_channel_message.subject == "update game phase") {
535             std::cout « "Update game phase message received by " « this « std::endl;
536 
537             if (
538                 game_channel_message.string_payload["game phase"] == "system management"
539             ) {
540                 this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
541                 this->__advanceTurn();
542             }
543 
544             else if (
545                 game_channel_message.string_payload["game phase"] == "loss emissions"
546             ) {
547                 this->game_phase = GamePhase :: LOSS_EMISSIONS;
548             }
549 
550             else if (
551                 game_channel_message.string_payload["game phase"] == "loss demand"
552             ) {
553                 this->game_phase = GamePhase :: LOSS_DEMAND;
554             }
555 
556             else if (
557                 game_channel_message.string_payload["game phase"] == "loss credits"
558             ) {
559                 this->game_phase = GamePhase :: LOSS_CREDITS;
560             }
561 
562             else if (
563                 game_channel_message.string_payload["game phase"] == "victory"
```

```
564             ) {
565                 this->game_phase = GamePhase :: VICTORY;
566             }
567
568             this->message_hub.popMessage(GAME_CHANNEL);
569         }
570
571         if (game_channel_message.subject == "improvement state") {
572             std::cout « "Improvement state message received by " « this « std::endl;
573
574             this->__handleImprovementStateMessage(game_channel_message);
575
576             this->message_hub.popMessage(GAME_CHANNEL);
577         }
578     }
579
580     if (not this->message_hub.isEmpty(GAME_STATE_CHANNEL)) {
581         Message game_state_message =
582             this->message_hub.receiveMessage(GAME_STATE_CHANNEL);
583
584         if (game_state_message.subject == "turn advance") {
585             std::cout « "Turn advance message received by " « this « std::endl;
586             this->message_hub.popMessage(GAME_STATE_CHANNEL);
587         }
588     }
589
590     return;
591 }   /* __processMessage() */
```

4.5.3.13 __sendCreditsEarnedMessage()

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

Helper method to format and send a credits earned message.

```
461 {
462     Message credits_earned_message;
463
464     credits_earned_message.channel = SETTLEMENT_CHANNEL;
465     credits_earned_message.subject = "credits earned";
466
467     this->message_hub.sendMessage(credits_earned_message);
468
469     std::cout « "Credits earned message sent by " « this « std::endl;
470     return;
471 }   /* __sendCreditsEarnedMessage() */
```

4.5.3.14 __sendGameStateMessage()

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

Helper method to format and send a game state message.

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

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

4.5.3.15 __sendTurnAdvanceMessage()

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

Helper method to format and send a turn advance message.

```
436 {
437     Message turn_advance_message;
438 
439     turn_advance_message.channel = GAME_STATE_CHANNEL;
440     turn_advance_message.subject = "turn advance";
441 
442     this->message_hub.sendMessage(turn_advance_message);
443 
444     std::cout « "Turn advance message sent by " « this « std::endl;
445     return;
446 }   /* __sendTurnAdvanceMessage() */
```

4.5.3.16 __toggleFrameClockOverlay()

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

Helper method to toggle frame clock overlay.

```
68 {
69     if (this->show_frame_clock_overlay) {
70         this->show_frame_clock_overlay = false;
71     }
72 
73     else {
74         this->show_frame_clock_overlay = true;
75     }
76 
77     return;
78 }   /* __toggleFrameClockOverlay() */
```

4.5.3.17 run()

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

Method to run game (defines game loop).

Returns

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

```
999 {
1000     //  1. play brand animation
1001     //...
1002 
1003     //  2. show splash screen
1004     //...
1005 
1006     //  3. start game loop
1007     while (not this->game_loop_broken) {
1008         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
1009 
1010         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
1011             //  6.1. process events
1012             while (this->render_window_ptr->pollEvent(this->event)) {
1013                 this->hex_map_ptr->processEvent();
1014                 this->context_menu_ptr->processEvent();
1015                 this->__processEvent();
1016             }
1017 
1018 
1019             //  6.2. process messages
1020             while (this->message_hub.hasTraffic()) {
1021                 this->hex_map_ptr->processMessage();
1022                 this->context_menu_ptr->processMessage();
1023                 this->__processMessage();
1024 
1025                 this->check_terminating_conditions = true;
1026 
1027                 if (not this->message_deadlock) {
1028                     this->message_deadlock_frame++;
1029 
1030                     if (this->message_deadlock_frame > 5 * FRAMES_PER_SECOND) {
1031                         this->message_hub.printState();
1032                         this->message_deadlock = true;
1033                     }
1034                 }
1035             }
1036             this->message_deadlock = false;
1037             this->message_deadlock_frame = 0;
1038 
1039 
1040             //  6.3. check terminating conditions
1041             if (this->check_terminating_conditions) {
1042                 this->__checkTerminatingConditions();
```

```
1043                 this->check_terminating_conditions = false;
1044             }
1045 
1046 
1047             //  6.4. draw frame
1048             this->render_window_ptr->clear();
1049 
1050             this->hex_map_ptr->draw();
1051             this->context_menu_ptr->draw();
1052             this->__draw();
1053 
1054             this->render_window_ptr->display();
1055 
1056 
1057             //  6.5. increment frame
1058             this->frame++;
1059         }
1060 
1061         // check track status, move to next if stopped
1062         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
1063             this->assets_manager_ptr->nextTrack();
1064             this->assets_manager_ptr->playTrack();
1065         }
1066 
1067     }
1068 
1069     return this->quit_game;
1070 }   /* run() */
```

4.5.4 Member Data Documentation

4.5.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.5.4.2 check_terminating_conditions

```
bool Game::check_terminating_conditions
```

Boolean indicating whether or not to check terminating conditions.

4.5.4.3 clock

```
sf::Clock Game::clock
```

The game clock.

4.5.4.4 context_menu_ptr

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

Pointer to the context menu.

4.5.4.5 credits

```
int Game::credits
```

Current balance of credits.

4.5.4.6 cumulative_emissions_tonnes

```
int Game::cumulative_emissions_tonnes
```

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

4.5.4.7 demand_MWh

```
int Game::demand_MWh
```

Current energy demand [MWh].

4.5.4.8 demand_remaining_MWh

```
int Game::demand_remaining_MWh
```

The current remaining energy demand [MWh].

4.5.4.9 demand_vec_MWh

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

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

4.5.4.10 event

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

The game events class.

4.5.4.11 frame

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

The current frame of the game.

4.5.4.12 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.13 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.14 hex_map_ptr

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

Pointer to the hex map (defines game world).

4.5.4.15 message_deadlock

```
bool Game::message_deadlock
```

A boolean indicating whether a message deadlock has been detected.

4.5.4.16 message_deadlock_frame

```
int Game::message_deadlock_frame
```

A frame counter for detecting message deadlock.

4.5.4.17 message_hub

```
MessageHub Game::message_hub
```

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

4.5.4.18 month

```
int Game::month
```

Current game month.

4.5.4.19 population

```
int Game::population
```

Current population.

4.5.4.20 quit_game

```
bool Game::quit_game
```

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

4.5.4.21 render_window_ptr

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

A pointer to the render window.

4.5.4.22 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

4.5.4.23 time_since_start_s

```
double Game::time_since_start_s
```

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

4.5.4.24 turn

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

The current game turn.

4.5.4.25 year

```
int Game::year
```

Current game year.

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

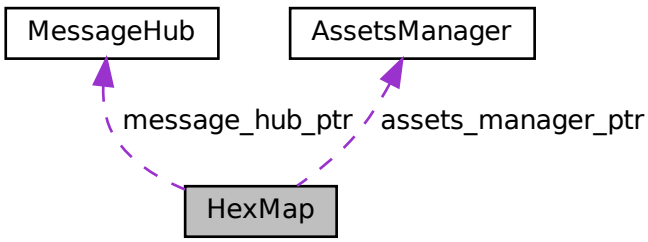
- header/Game.h
- source/Game.cpp

4.6 HexMap Class Reference

A class which defines a hex map of hex tiles.

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

Collaboration diagram for HexMap:



Public Member Functions

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

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

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

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

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

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

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

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

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

  Destructor for the HexMap class.

Public Attributes

- bool show_resource

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

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

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

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

  The current frame of this object.
- double position_x

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

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

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

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

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

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

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

  A vector of hex tiles, in drawing order.

Private Member Functions

- void __setUpGlassScreen (void)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  Helper method to assess all neighbours of the given tile.

Private Attributes

- sf::Event * event_ptr

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

  A pointer to the render window.
- AssetsManager * assets_manager_ptr

  A pointer to the assets manager.
- MessageHub * message_hub_ptr

  A pointer to the message hub.

4.6.1 Detailed Description

A class which defines a hex map of hex tiles.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 HexMap()

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

Constructor (intended) for the HexMap class.

Parameters

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.

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

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

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

Helper method to assemble the hex map.

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

4.6.3.2 __assessNeighbours()

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

Helper method to assess all neighbours of the given tile.

Parameters

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.

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

4.6.3.21 draw()

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

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

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

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

4.6.3.22 processEvent()

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

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

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

4.6.3.23 processMessage()

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

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

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

4.6.3.24 reroll()

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

Method to re-roll the hex map.

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

4.6.3.25 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

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

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

4.6.4 Member Data Documentation

4.6.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.6.4.2 border_tiles_vec

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

A vector of pointers to the border tiles.

4.6.4.3 event_ptr

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

A pointer to the event class.

4.6.4.4 frame

```
unsigned long long int HexMap::frame
```

The current frame of this object.

4.6.4.5 glass_screen

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

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

4.6.4.6 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

4.6.4.7 hex_map

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

A position-indexed, nested map of hex tiles.

4.6.4.8 message_hub_ptr

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

A pointer to the message hub.

4.6.4.9 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

4.6.4.10 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

4.6.4.11 position_x

```
double HexMap::position_x
```

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

4.6.4.12 position_y

```
double HexMap::position_y
```

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

4.6.4.13 render_window_ptr

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

A pointer to the render window.

4.6.4.14 show_resource

```
bool HexMap::show_resource
```

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

4.6.4.15 tile_position_x_vec

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

A vector of tile x positions.

4.6.4.16 tile_position_y_vec

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

A vector of tile y position.

4.6.4.17 tile_selected

```
bool HexMap::tile_selected
```

A boolean which indicates if a tile is currently selected.

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

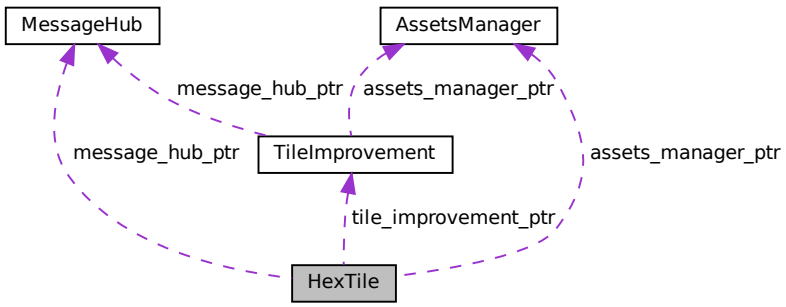
- header/HexMap.h
- source/HexMap.cpp

4.7 HexTile Class Reference

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

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

Collaboration diagram for HexTile:



Public Member Functions

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

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

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

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

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

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

  Method to decorate tile.
- void toggleResourceOverlay (void)

  Method to toggle the tile resource overlay.

- void assess (void)

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

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

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

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

  Destructor for the HexTile class.

Public Attributes

- TileType tile_type

  The terrain type of the tile.
- TileResource tile_resource

  The renewable resource quality of the tile.
- bool show_node

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

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

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

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

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

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

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

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

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

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

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

  The current frame of this object.
- int credits

  The current balance of credits.
- int scrap_improvement_frame

  A frame for key-hold to confirm scrapping.
- double position_x

  The x position of the tile.
- double position_y

  The y position of the tile.
- double major_radius

The radius of the smallest bounding circle.

- double minor_radius

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

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

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

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

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

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

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

  A tile decoration sprite.
- sf::Sprite magnifying_glass_sprite

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

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

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

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

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

  A vector of text for the tile build options.

Private Member Functions

- void __setUpNodeSprite (void)

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

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

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

  Helper method to set up resource chip sprite.
- void __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 __handleKeyReleaseEvents (void)
- void __handleMouseButtonEvents (void)

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

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

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

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

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

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

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

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

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

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

  Helper method to scrap the tile improvement (Settlement cannot be scrapped). Requires the mapped key to be held continuously to confirm.
- void __sendTileSelectedMessage (void)

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

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

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

Helper method to assemble and return tile resource substring.

- std::string __getTileImprovementSubstring (void)

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

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

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

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

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

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

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

  Helper method to format and send an insufficient credits message.

Private Attributes

- sf::Event * event_ptr

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

  A pointer to the render window.
- AssetsManager * assets_manager_ptr

  A pointer to the assets manager.
- MessageHub * message_hub_ptr

  A pointer to the message hub.

4.7.1 Detailed Description

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

4.7.2 Constructor & Destructor Documentation

4.7.2.1 HexTile()

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

Constructor for the HexTile class.

Ref: Wikipedia [2023]

Parameters

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.

```
2309 {
2310     //  1. set attributes
2311 
2312     //  1.1. private
2313     this->event_ptr = event_ptr;
2314     this->render_window_ptr = render_window_ptr;
2315 
2316     this->assets_manager_ptr = assets_manager_ptr;
2317     this->message_hub_ptr = message_hub_ptr;
2318 
2319     //  1.2. public
2320     this->show_node = false;
2321     this->show_resource = false;
2322     this->resource_assessed = false;
2323     this->resource_assessment = false;
2324     this->is_selected = false;
2325     this->draw_explosion = false;
2326 
2327     this->decoration_cleared = false;
2328     this->has_improvement = false;
2329     this->tile_improvement_ptr = NULL;
2330 
2331     this->build_menu_open = false;
2332 
2333     this->explosion_frame = 0;
2334 
2335     this->frame = 0;
2336     this->credits = 0;
2337 
2338     this->scrap_improvement_frame = 0;
2339 
2340     this->position_x = position_x;
2341     this->position_y = position_y;
2342 
2343     this->major_radius = 32;
2344     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2345 
2346     this->game_phase = "build settlement";
2347 
2348     //  2. set up and position drawable attributes
2349     this->__setUpNodeSprite();
2350     this->__setUpTileSprite();
2351     this->__setUpSelectOutlineSprite();
2352     this->__setUpResourceChipSprite();
2353     this->__setResourceText();
2354     this->__setUpMagnifyingGlassSprite();
2355     this->__setUpTileExplosionReel();
2356 
2357     //  3. set tile type and resource (default to none type and average)
2358     this->setTileType(TileType :: NONE_TYPE);
2359     this->setTileResource(TileResource :: AVERAGE);
2360 
2361     std::cout « "HexTile constructed at " « this « std::endl;
2362 
2363     return;
2364 }   /* HexTile() */
```

4.7.2.2 ~HexTile()

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

Destructor for the HexTile class.

```
2927 {
2928     if (this->tile_improvement_ptr != NULL) {
2929         delete this->tile_improvement_ptr;
2930     }
2931
2932     std::cout « "HexTile at " « this « " destroyed" « std::endl;
2933
2934     return;
2935 }   /* ~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.

```
1410 {
1411     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1412
1413     if (this->credits < build_cost) {
1414         std::cout « "Cannot build diesel generator: insufficient credits (need "
1415             « build_cost « " K)" « std::endl;
1416
1417         this->__sendInsufficientCreditsMessage();
1418         return;
1419     }
1420
1421     this->tile_improvement_ptr = new DieselGenerator(
1422         this->position_x,
1423         this->position_y,
1424         this->tile_resource,
1425         this->event_ptr,
1426         this->render_window_ptr,
1427         this->assets_manager_ptr,
1428         this->message_hub_ptr
1429     );
1430
1431     this->has_improvement = true;
1432     this->__closeBuildMenu();
1433
1434     this->__sendCreditsSpentMessage(build_cost);
1435     this->__sendTileStateMessage();
1436     this->__sendGameStateRequest();
1437
1438     return;
1439 }   /* __buildDieselGenerator() */
```

4.7.3.2 __buildEnergyStorage()

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

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

```
1658 {
1659     /*
1660     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1661
1662     if (this->credits < build_cost) {
1663         std::cout « "Cannot build energy storage system: insufficient credits (need "
1664             « build_cost « " K)" « std::endl;
1665
1666         this->__sendInsufficientCreditsMessage();
1667         return;
```

```
1668     }
1669
1670     this->tile_improvement_ptr = new EnergyStorageSystem(
1671         this->position_x,
1672         this->position_y,
1673         this->event_ptr,
1674         this->render_window_ptr,
1675         this->assets_manager_ptr,
1676         this->message_hub_ptr
1677     );
1678
1679     this->has_improvement = true;
1680     this->__closeBuildMenu();
1681
1682     this->__sendCreditsSpentMessage(build_cost);
1683     this->__sendTileStateMessage();
1684     this->__sendGameStateRequest();
1685     */
1686     return;
1687 }   /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

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

Helper method to build a settlement on this tile.

```
1363 {
1364     if (this->credits < BUILD_SETTLEMENT_COST) {
1365         std::cout « "Cannot build settlement: insufficient credits (need "
1366             « BUILD_SETTLEMENT_COST « " K)" « std::endl;
1367
1368         this->__sendInsufficientCreditsMessage();
1369         return;
1370     }
1371
1372     this->__clearDecoration();
1373
1374     this->tile_improvement_ptr = new Settlement(
1375         this->position_x,
1376         this->position_y,
1377         this->tile_resource,
1378         this->event_ptr,
1379         this->render_window_ptr,
1380         this->assets_manager_ptr,
1381         this->message_hub_ptr
1382     );
1383
1384     this->has_improvement = true;
1385
1386     this->assess();
1387     this->__sendAssessNeighboursMessage();
1388
1389     this->__sendUpdateGamePhaseMessage("system management");
1390     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1391     this->__sendTileStateMessage();
1392     this->__sendGameStateRequest();
1393
1394     return;
1395 }   /* __buildSettlement() */
```

4.7.3.4 __buildSolarPV()

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

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

```
1454 {
1455     int build_cost = SOLAR_PV_BUILD_COST;
```

```
1456
1457     if (this->tile_type == TileType :: LAKE) {
1458         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1459     }
1460
1461     if (this->credits < build_cost) {
1462         std::cout « "Cannot build solar PV array: insufficient credits (need "
1463             « build_cost « " K)" « std::endl;
1464
1465         this->__sendInsufficientCreditsMessage();
1466         return;
1467     }
1468
1469     this->tile_improvement_ptr = new SolarPV(
1470         this->position_x,
1471         this->position_y,
1472         this->tile_resource,
1473         this->event_ptr,
1474         this->render_window_ptr,
1475         this->assets_manager_ptr,
1476         this->message_hub_ptr
1477     );
1478
1479     this->has_improvement = true;
1480     this->__closeBuildMenu();
1481
1482     if (this->tile_type == TileType :: LAKE) {
1483         this->decoration_cleared = true;
1484         this->assets_manager_ptr->getSound("splash")->play();
1485     }
1486
1487     this->__sendCreditsSpentMessage(build_cost);
1488     this->__sendTileStateMessage();
1489     this->__sendGameStateRequest();
1490
1491     return;
1492 }   /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

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

Helper method to build a tidal turbine on this tile.

```
1566 {
1567     int build_cost = TIDAL_TURBINE_BUILD_COST;
1568
1569     if (this->credits < build_cost) {
1570         std::cout « "Cannot build tidal turbine: insufficient credits (need "
1571             « build_cost « " K)" « std::endl;
1572
1573         this->__sendInsufficientCreditsMessage();
1574         return;
1575     }
1576
1577     this->tile_improvement_ptr = new TidalTurbine(
1578         this->position_x,
1579         this->position_y,
1580         this->tile_resource,
1581         this->event_ptr,
1582         this->render_window_ptr,
1583         this->assets_manager_ptr,
1584         this->message_hub_ptr
1585     );
1586
1587     this->has_improvement = true;
1588     this->decoration_cleared = true;
1589     this->assets_manager_ptr->getSound("splash")->play();
1590     this->__closeBuildMenu();
1591
1592     this->__sendCreditsSpentMessage(build_cost);
1593     this->__sendTileStateMessage();
1594     this->__sendGameStateRequest();
1595
1596     return;
1597 }   /* __buildTidalTurbine() */
```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1612 {
1613     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1614
1615     if (this->credits < build_cost) {
1616         std::cout « "Cannot build wave energy converter: insufficient credits (need "
1617             « build_cost « " K)" « std::endl;
1618
1619         this->__sendInsufficientCreditsMessage();
1620         return;
1621     }
1622
1623     this->tile_improvement_ptr = new WaveEnergyConverter(
1624         this->position_x,
1625         this->position_y,
1626         this->tile_resource,
1627         this->event_ptr,
1628         this->render_window_ptr,
1629         this->assets_manager_ptr,
1630         this->message_hub_ptr
1631     );
1632
1633     this->has_improvement = true;
1634     this->decoration_cleared = true;
1635     this->assets_manager_ptr->getSound("splash")->play();
1636     this->__closeBuildMenu();
1637
1638     this->__sendCreditsSpentMessage(build_cost);
1639     this->__sendTileStateMessage();
1640     this->__sendGameStateRequest();
1641
1642     return;
1643 }   /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1507 {
1508     int build_cost = WIND_TURBINE_BUILD_COST;
1509
1510     if (
1511         (this->tile_type == TileType :: LAKE) or
1512         (this->tile_type == TileType :: OCEAN)
1513     ) {
1514         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1515     }
1516
1517     if (this->credits < build_cost) {
1518         std::cout « "Cannot build wind turbine: insufficient credits (need "
1519             « build_cost « " K)" « std::endl;
1520
1521         this->__sendInsufficientCreditsMessage();
1522         return;
1523     }
1524
1525     this->tile_improvement_ptr = new WindTurbine(
1526         this->position_x,
1527         this->position_y,
1528         this->tile_resource,
1529         this->event_ptr,
1530         this->render_window_ptr,
1531         this->assets_manager_ptr,
1532         this->message_hub_ptr
1533     );
1534
1535     this->has_improvement = true;
1536     this->__closeBuildMenu();
```

```
1537
1538     if (
1539         (this->tile_type == TileType :: LAKE) or
1540         (this->tile_type == TileType :: OCEAN)
1541     ) {
1542         this->decoration_cleared = true;
1543         this->assets_manager_ptr->getSound("splash")->play();
1544     }
1545
1546     this->__sendCreditsSpentMessage(build_cost);
1547     this->__sendTileStateMessage();
1548     this->__sendGameStateRequest();
1549
1550     return;
1551 }   /* __buildWindTurbine() */
```

4.7.3.8 __clearDecoration()

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

Helper method to clear tile decoration.

```
791 {
792     this->decoration_cleared = true;
793     this->draw_explosion = true;
794
795     switch (this->tile_type) {
796         case (TileType :: FOREST): {
797             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
798
799             break;
800         }
801
802
803         case (TileType :: MOUNTAINS): {
804             this->assets_manager_ptr->getSound("clear mountains tile")->play();
805
806             break;
807         }
808
809
810         case (TileType :: PLAINS): {
811             this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
812
813             break;
814         }
815
816
817         default: {
818             // do nothing!
819
820             break;
821         }
822     }
823
824     return;
825 }   /* __clearDecoration() */
```

4.7.3.9 __closeBuildMenu()

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

Helper method to close the tile improvement build menu.

```
1338 {
1339     if (not this->build_menu_open) {
1340         return;
1341     }
1342
1343     this->build_menu_open = false;
1344     this->assets_manager_ptr->getSound("build menu close")->play();
1345
1346     return;
1347 }   /* __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.

```
1804 {
1805     std::string coords_substring = "TILE COORDS:  (";
1806     coords_substring += std::to_string(int(this->position_x - 400));
1807     coords_substring += ", ";
1808     coords_substring += std::to_string(int(this->position_y - 400));
1809     coords_substring += ")\n";
1810 
1811     return coords_substring;
1812 }   /* __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.

```
1963 {
1964     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1965 
1966     if (this->has_improvement) {
1967         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1968         improvement_substring += "\n";
1969     }
1970 
1971     else {
1972         improvement_substring += "NONE\n";
1973     }
1974 
1975     return improvement_substring;
1976 }   /* __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.

```
1993 {
1994     //                  32 char x 17 line console "--------------------------------\n";
1995     std::string options_substring                = "     **** TILE OPTIONS ****     \n";
1996     options_substring                           += "                                \n";
1997
1998     if (this->game_phase == "build settlement") {
1999         if (
2000             (this->tile_type != TileType :: OCEAN) and
2001             (this->tile_type != TileType :: LAKE)
2002         ) {
2003             options_substring += "[B]:  BUILD SETTLEMENT (";
2004             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
2005             options_substring += " K)\n";
2006         }
2007     }
2008
2009
2010     else if (this->game_phase == "system management") {
2011         if (this->has_improvement) {
2012             options_substring.clear();
2013             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2014         }
2015
2016
2017         else if (not this->resource_assessed) {
2018             options_substring += "[A]:  ASSESS RESOURCE (";
2019             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2020             options_substring += " K)\n";
2021         }
2022
2023
2024         else if (
2025             (not this->decoration_cleared) and
2026             (this->tile_type != TileType :: OCEAN) and
2027             (this->tile_type != TileType :: LAKE)
2028         ) {
2029             options_substring += "[C]:  CLEAR TILE (";
2030
2031             switch (this->tile_type) {
2032                 case (TileType :: FOREST): {
2033                     options_substring += std::to_string(CLEAR_FOREST_COST);
2034
2035                     break;
2036                 }
2037
2038
2039                 case (TileType :: MOUNTAINS): {
2040                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2041
2042                     break;
2043                 }
2044
2045
2046                 case (TileType :: PLAINS): {
2047                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2048
2049                     break;
2050                 }
2051
2052
2053                 default: {
2054                     //do nothing!
2055
2056                     break;
2057                 }
2058             }
2059
2060             options_substring += " K)\n";
2061         }
2062
2063
2064         else if (
2065             (this->decoration_cleared) or
2066             (this->tile_type == TileType :: OCEAN) or
2067             (this->tile_type == TileType :: LAKE)
2068         ) {
2069             options_substring += "[B]: OPEN BUILD MENU\n";
2070         }
2071     }
2072
2073
2074     else if (this->game_phase == "victory") {
2075         options_substring                       += "       **** VICTORY ****        \n";
2076     }
```

```
2077
2078
2079     else {
2080         options_substring                       += "        **** LOSS ****          \n";
2081     }
2082
2083     return options_substring;
2084 }   /* __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.

```
1893 {
1894     std::string resource_substring = "TILE RESOURCE:     ";
1895
1896     if (this->resource_assessed) {
1897         switch (this->tile_resource) {
1898             case (TileResource :: POOR): {
1899                 resource_substring += "POOR\n";
1900
1901                 break;
1902             }
1903
1904
1905             case (TileResource ::BELOW_AVERAGE): {
1906                 resource_substring += "BELOW AVERAGE\n";
1907
1908                 break;
1909             }
1910
1911
1912             case (TileResource :: AVERAGE): {
1913                 resource_substring += "AVERAGE\n";
1914
1915                 break;
1916             }
1917
1918
1919             case (TileResource :: ABOVE_AVERAGE): {
1920                 resource_substring += "ABOVE AVERAGE\n";
1921
1922                 break;
1923             }
1924
1925
1926             case (TileResource :: GOOD): {
1927                 resource_substring += "GOOD\n";
1928
1929                 break;
1930             }
1931
1932
1933             default: {
1934                 resource_substring += "???\n";
1935
1936                 break;
1937             }
1938         }
1939     }
1940
1941     else {
1942         resource_substring += "???\n";
1943     }
1944
1945     return resource_substring;
1946 }   /* __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.

```
1829 {
1830     std::string type_substring = "TILE TYPE:         ";
1831 
1832     switch (this->tile_type) {
1833         case (TileType :: FOREST): {
1834             type_substring += "FOREST\n";
1835 
1836             break;
1837         }
1838 
1839 
1840         case (TileType :: LAKE): {
1841             type_substring += "LAKE\n";
1842 
1843             break;
1844         }
1845 
1846 
1847         case (TileType :: MOUNTAINS): {
1848             type_substring += "MOUNTAINS\n";
1849 
1850             break;
1851         }
1852 
1853 
1854         case (TileType :: OCEAN): {
1855             type_substring += "OCEAN\n";
1856 
1857             break;
1858         }
1859 
1860 
1861         case (TileType :: PLAINS): {
1862             type_substring += "PLAINS\n";
1863 
1864             break;
1865         }
1866 
1867 
1868         default: {
1869             type_substring += "???\n";
1870 
1871             break;
1872         }
1873     }
1874 
1875     return type_substring;
1876 }   /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
874 {
875     if (not this->is_selected) {
876         return;
877     }
878 
879 
880     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
```

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

```
968                     }
969
970
971                     case (sf::Keyboard::W): {
972                         this->__buildWindTurbine();
973
974                         break;
975                     }
976
977
978                     case (sf::Keyboard::E): {
979                         this->__buildEnergyStorage();
980
981                         break;
982                     }
983
984
985                     default: {
986                         // do nothing!
987
988                         break;
989                     }
990                 }
991
992                 break;
993             }
994
995
996             case (TileType :: OCEAN): {
997                 switch (this->event_ptr->key.code) {
998                     case (sf::Keyboard::W): {
999                         this->__buildWindTurbine();
1000
1001                         break;
1002                     }
1003
1004
1005                     case (sf::Keyboard::T): {
1006                         this->__buildTidalTurbine();
1007
1008                         break;
1009                     }
1010
1011
1012                     case (sf::Keyboard::A): {
1013                         this->__buildWaveEnergyConverter();
1014
1015                         break;
1016                     }
1017
1018
1019                     default: {
1020                         // do nothing!
1021
1022                         break;
1023                     }
1024                 }
1025
1026                 break;
1027             }
1028
1029
1030             case (TileType :: PLAINS): {
1031                 switch (this->event_ptr->key.code) {
1032                     case (sf::Keyboard::D): {
1033                         this->__buildDieselGenerator();
1034
1035                         break;
1036                     }
1037
1038
1039                     case (sf::Keyboard::S): {
1040                         this->__buildSolarPV();
1041
1042                         break;
1043                     }
1044
1045
1046                     case (sf::Keyboard::W): {
1047                         this->__buildWindTurbine();
1048
1049                         break;
1050                     }
1051
1052
1053                     case (sf::Keyboard::E): {
1054                         this->__buildEnergyStorage();
```

```
1055
1056                        break;
1057                    }
1058
1059
1060                    default: {
1061                        // do nothing!
1062
1063                        break;
1064                    }
1065                }
1066
1067                break;
1068            }
1069
1070
1071            default: {
1072                //do nothing!
1073
1074                break;
1075            }
1076        }
1077    }
1078
1079
1080    if (this->game_phase == "build settlement") {
1081        if (
1082            (this->tile_type != TileType :: OCEAN) and
1083            (this->tile_type != TileType :: LAKE)
1084        ) {
1085            if (this->event_ptr->key.code == sf::Keyboard::B) {
1086                this->__buildSettlement();
1087            }
1088        }
1089    }
1090
1091
1092    else if (this->game_phase == "system management") {
1093        if (this->has_improvement) {
1094            if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
    {
1095                if (this->event_ptr->key.code == sf::Keyboard::P) {
1096                    this->__scrapImprovement();
1097                }
1098            }
1099
1100            /*
1101             * All other inputs will be caught and handled by
1102             *   this->tile_improvement_ptr->processEvent()
1103             */
1104        }
1105
1106
1107        else if (not this->resource_assessed) {
1108            if (this->event_ptr->key.code == sf::Keyboard::A) {
1109                if (this->credits < RESOURCE_ASSESSMENT_COST) {
1110                    std::cout « "Cannot assess resource: insufficient credits (need "
1111                        « RESOURCE_ASSESSMENT_COST « " K)" « std::endl;
1112
1113                    this->__sendInsufficientCreditsMessage();
1114                }
1115
1116                else {
1117                    this->assess();
1118                    this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1119                    this->__sendTileStateMessage();
1120                    this->__sendGameStateRequest();
1121                }
1122            }
1123        }
1124
1125
1126        else if (
1127            (not this->decoration_cleared) and
1128            (this->tile_type != TileType :: OCEAN) and
1129            (this->tile_type != TileType :: LAKE)
1130        ) {
1131            if (this->event_ptr->key.code == sf::Keyboard::C) {
1132                int clear_cost = 0;
1133
1134                switch (this->tile_type) {
1135                    case (TileType :: FOREST): {
1136                        clear_cost = CLEAR_FOREST_COST;
1137
1138                        break;
1139                    }
1140
```

```
1141
1142                     case (TileType :: MOUNTAINS): {
1143                         clear_cost = CLEAR_MOUNTAINS_COST;
1144
1145                         break;
1146                     }
1147
1148
1149                     case (TileType :: PLAINS): {
1150                         clear_cost = CLEAR_PLAINS_COST;
1151
1152                         break;
1153                     }
1154
1155
1156                     default: {
1157                         // do nothing!
1158
1159                         break;
1160                     }
1161                 }
1162
1163                 if (this->credits < clear_cost) {
1164                     std::cout « "Cannot clear tile: insufficient credits (need "
1165                         « clear_cost « " K)" « std::endl;
1166
1167                     this->__sendInsufficientCreditsMessage();
1168                 }
1169
1170                 else {
1171                     this->__clearDecoration();
1172                     this->__sendCreditsSpentMessage(clear_cost);
1173                     this->__sendTileStateMessage();
1174                     this->__sendGameStateRequest();
1175                 }
1176             }
1177         }
1178
1179
1180         else if (
1181             (this->decoration_cleared) or
1182             (this->tile_type == TileType :: OCEAN) or
1183             (this->tile_type == TileType :: LAKE)
1184         ) {
1185             if (this->event_ptr->key.code == sf::Keyboard::B) {
1186                 this->__openBuildMenu();
1187             }
1188         }
1189     }
1190
1191     return;
1192 }   /* __handleKeyPressEvents() */
```

4.7.3.16 __handleKeyReleaseEvents()

```
void HexTile::__handleKeyReleaseEvents (
             void  )  [private]
1198 {
1199     if (not this->is_selected) {
1200         return;
1201     }
1202
1203
1204     switch (this->event_ptr->key.code) {
1205         case (sf::Keyboard::P): {
1206             if (this->has_improvement) {
1207                 this->scrap_improvement_frame = 0;
1208
1209                 if (
1210                     this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1211                 ) {
1212                     this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1213                         sf::Color(255, 255, 255, 255)
1214                     );
1215                 }
1216
1217                 else {
1218                     for (
1219                         size_t i = 0;
```

```
1220                         i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1221                         i++
1222                     ) {
1223                         this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1224                             sf::Color(255, 255, 255, 255)
1225                         );
1226                     }
1227                 }
1228             }
1229
1230
1231             break;
1232         }
1233
1234
1235         default: {
1236             // do nothing!
1237
1238             break;
1239         }
1240     }
1241
1242     /*
1243     if (this->event_ptr->key.code == sf::Keyboard::P) {
1244
1245     }
1246     */
1247
1248     return;
1249 }   /* __handleKeyReleaseEvents() */
```

4.7.3.17 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
1262 {
1263     switch (this->event_ptr->mouseButton.button) {
1264         case (sf::Mouse::Left): {
1265             if (this->__isClicked()) {
1266                 std::cout « "Tile (" « this->position_x « ", " «
1267                     this->position_y « ") was selected" « std::endl;
1268
1269                 this->__setIsSelected(true);
1270
1271                 this->__sendTileSelectedMessage();
1272                 this->__sendTileStateMessage();
1273                 this->__sendGameStateRequest();
1274             }
1275
1276             else {
1277                 this->__setIsSelected(false);
1278             }
1279
1280             break;
1281         }
1282
1283
1284         case (sf::Mouse::Right): {
1285             this->__setIsSelected(false);
1286
1287             break;
1288         }
1289
1290
1291         default: {
1292             // do nothing!
1293
1294             break;
1295         }
1296     }
1297
1298     return;
1299 }   /* __handleMouseButtonEvents() */
```

4.7.3.18 __isClicked()

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

Helper method to determine if tile was clicked on.

Returns

Boolean indicating whether or not tile was clicked on.

```
842 {
843     sf::Vector2i mouse_position = sf::Mouse::getPosition(*render_window_ptr);
844 
845     double mouse_x = mouse_position.x;
846     double mouse_y = mouse_position.y;
847 
848     double distance = sqrt(
849         pow(this->position_x - mouse_x, 2) +
850         pow(this->position_y - mouse_y, 2)
851     );
852 
853     if (distance < this->minor_radius) {
854         return true;
855     }
856     else {
857         return false;
858     }
859 }   /* __isClicked() */
```

4.7.3.19 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

```
1314 {
1315     if (this->build_menu_open) {
1316         return;
1317     }
1318 
1319     this->build_menu_open = true;
1320     this->assets_manager_ptr->getSound("build menu open")->play();
1321 
1322     return;
1323 }   /* __openBuildMenu() */
```

4.7.3.20 __scrapImprovement()

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

Helper method to scrap the tile improvement (Settlement cannot be scrapped). Requires the mapped key to be held continuously to confirm.

```
1703 {
1704     //  1. implement key hold confirmation
1705     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1706         double colour_scalar =
1707             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1708 
1709         if (
1710             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1711         ) {
```

```
1712             this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1713                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1714             );
1715         }
1716 
1717         else {
1718             for (
1719                 size_t i = 0;
1720                 i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1721                 i++
1722             ) {
1723                 this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1724                     sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1725                 );
1726             }
1727         }
1728 
1729         this->scrap_improvement_frame += 4;
1730     }
1731 
1732 
1733     //  2. carry out scrapping
1734     else {
1735         this->draw_explosion = true;
1736         this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1737 
1738         if (this->tile_improvement_ptr->production_menu_open) {
1739             this->tile_improvement_ptr->production_menu_open = false;
1740             this->assets_manager_ptr->getSound("build menu close")->play();
1741         }
1742 
1743         delete this->tile_improvement_ptr;
1744         this->tile_improvement_ptr = NULL;
1745 
1746         this->has_improvement = false;
1747 
1748         this->scrap_improvement_frame = 0;
1749 
1750         if (
1751             (this->tile_type == TileType :: LAKE) or
1752             (this->tile_type == TileType :: OCEAN)
1753         ) {
1754             this->decoration_cleared = false;
1755         }
1756 
1757         this->__sendCreditsSpentMessage(SCRAP_COST);
1758         this->__sendTileStateMessage();
1759         this->__sendGameStateRequest();
1760     }
1761 
1762     return;
1763 }   /* __scrapImprovement() */
```

4.7.3.21 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2140 {
2141     Message assess_neighbours_message;
2142 
2143     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2144     assess_neighbours_message.subject = "assess neighbours";
2145 
2146     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2147 
2148     std::cout « "Assess neighbours message sent by " « this « std::endl;
2149 
2150     return;
2151 }   /* __sendAssessNeighboursMessage() */
```

4.7.3.22 __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.

```
2223 {
2224     Message credits_spent_message;
2225 
2226     credits_spent_message.channel = GAME_CHANNEL;
2227     credits_spent_message.subject = "credits spent";
2228 
2229     credits_spent_message.int_payload["credits spent"] = credits_spent;
2230 
2231     this->message_hub_ptr->sendMessage(credits_spent_message);
2232 
2233     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2234         << std::endl;
2235     return;
2236 }   /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2166 {
2167     Message game_state_request;
2168 
2169     game_state_request.channel = GAME_CHANNEL;
2170     game_state_request.subject = "state request";
2171 
2172     this->message_hub_ptr->sendMessage(game_state_request);
2173 
2174     std::cout << "Game state request message sent by " << this << std::endl;
2175     return;
2176 }   /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2251 {
2252     Message insufficient_credits_message;
2253 
2254     insufficient_credits_message.channel = GAME_CHANNEL;
2255     insufficient_credits_message.subject = "insufficient credits";
2256 
2257     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2258 
2259     std::cout << "Insufficient credits message sent by " << this << std::endl;
2260 
2261     return;
2262 }   /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1778 {
1779     Message tile_selected_message;
1780 
1781     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1782     tile_selected_message.subject = "tile selected";
1783 
1784     this->message_hub_ptr->sendMessage(tile_selected_message);
1785 
1786     return;
1787 }   /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2099 {
2100     Message tile_state_message;
2101 
2102     tile_state_message.channel = TILE_STATE_CHANNEL;
2103     tile_state_message.subject = "tile state";
2104 
2105 
2106     //                  32 char x 17 line console "--------------------------------\n";
2107     std::string console_string                    = "      **** TILE INFO ****       \n";
2108 
2109     console_string                               += this->__getTileCoordsSubstring();
2110     console_string                               += "                                \n";
2111 
2112     console_string                               += this->__getTileTypeSubstring();
2113     console_string                               += this->__getTileResourceSubstring();
2114     console_string                               += this->__getTileImprovementSubstring();
2115     console_string                               += "                                \n";
2116 
2117     console_string                               += this->__getTileOptionsSubstring();
2118 
2119     tile_state_message.string_payload["console string"] = console_string;
2120 
2121     this->message_hub_ptr->sendMessage(tile_state_message);
2122 
2123     std::cout « "Tile state message sent by " « this « std::endl;
2124     return;
2125 }   /* __sendTileStateMessage() */
```

4.7.3.27 __sendUpdateGamePhaseMessage()

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

Helper method to format and send update game phase message.

Parameters

game_phase	The updated game phase.

```
2193 {
2194     Message update_game_phase_message;
2195 
2196     update_game_phase_message.channel = GAME_CHANNEL;
2197     update_game_phase_message.subject = "update game phase";
2198 
2199     update_game_phase_message.string_payload["game phase"] = game_phase;
2200 
2201     this->message_hub_ptr->sendMessage(update_game_phase_message);
2202 
2203     std::cout « "Update game phase message sent by " « this « std::endl;
2204 
2205     return;
2206 }   /* __sendUpdateGamePhaseMessage() */
```

4.7.3.28 __setIsSelected()

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

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

Parameters

is_selected	The value to set the is selected attribute to.

```
764 {
765     this->is_selected = is_selected;
766 
767     if (this->tile_improvement_ptr != NULL) {
768         this->tile_improvement_ptr->setIsSelected(is_selected);
769     }
770 
771     if ((not is_selected) and this->build_menu_open) {
772         this->__closeBuildMenu();
773     }
774 
775     return;
776 }   /* __setIsSelected() */
```

4.7.3.29 __setResourceText()

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

Helper method to set up resource text.

```
193 {
194     this->resource_text.setFont(*(assets_manager_ptr->getFont("DroidSansMono")));
195 
196     this->resource_text.setFillColor(sf::Color(0, 0, 0, 255));
197 
198     if (this->resource_assessed) {
199         switch (this->tile_resource) {
200             case (TileResource :: POOR): {
201                 this->resource_text.setString("-2");
202                 this->resource_text.setFillColor(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() */
```

4.7.3.30 __setUpBuildMenu()

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

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

```
667 {
668     this->build_menu_options_vec.clear();
669     this->build_menu_options_text_vec.clear();
670
671     //  1. set up and place build menu backing and text
672     this->build_menu_backing.setSize(sf::Vector2f(600, 256));
673     this->build_menu_backing.setOrigin(300, 128);
674     this->build_menu_backing.setPosition(400, 400);
675     this->build_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
676     this->build_menu_backing.setOutlineColor(MENU_FRAME_GREY);
677     this->build_menu_backing.setOutlineThickness(4);
678
679     this->build_menu_backing_text.setString("**** BUILD MENU ****");
680     this->build_menu_backing_text.setFont(
681         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
682     );
683     this->build_menu_backing_text.setCharacterSize(16);
684     this->build_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
685     this->build_menu_backing_text.setOrigin(
686         this->build_menu_backing_text.getLocalBounds().width / 2, 0
687     );
```

```
688     this->build_menu_backing_text.setPosition(400, 400 - 128 + 4);
689
690     //  2. set up and place build menu option sprites and text
691     switch (this->tile_type) {
692         case (TileType :: FOREST): {
693             this->__setUpDieselGeneratorBuildOption();
694             this->__setUpSolarPVBuildOption();
695             this->__setUpWindTurbineBuildOption();
696             //this->__setUpEnergyStorageSystemBuildOption();
697
698             break;
699         }
700
701
702         case (TileType :: LAKE): {
703             this->__setUpSolarPVBuildOption(true);
704             this->__setUpWindTurbineBuildOption(true);
705
706             break;
707         }
708
709
710         case (TileType :: MOUNTAINS): {
711             this->__setUpDieselGeneratorBuildOption();
712             this->__setUpSolarPVBuildOption();
713             this->__setUpWindTurbineBuildOption();
714             //this->__setUpEnergyStorageSystemBuildOption();
715
716             break;
717         }
718
719
720         case (TileType :: OCEAN): {
721             this->__setUpWindTurbineBuildOption(false, true);
722             this->__setUpTidalTurbineBuildOption();
723             this->__setUpWaveEnergyConverterBuildOption();
724
725             break;
726         }
727
728
729         case (TileType :: PLAINS): {
730             this->__setUpDieselGeneratorBuildOption();
731             this->__setUpSolarPVBuildOption();
732             this->__setUpWindTurbineBuildOption();
733             //this->__setUpEnergyStorageSystemBuildOption();
734
735             break;
736         }
737
738
739         default: {
740             // do nothing!
741
742             break;
743         }
744     }
745
746     return;
747 }   /* __setUpBuildMenu() */
```

4.7.3.31 __setUpBuildOption()

```
void HexTile::__setUpBuildOption (
            std::string texture_key,
            std::string option_string )  [private]
```

Helper method to set up and 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() */
```

4.7.3.32 __setUpDieselGeneratorBuildOption()

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

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

```
433 {
434     //  1. set up option sprite(s)
435     std::string texture_key = "diesel generator";
436 
437     //  2. set up option string (up to 16 chars wide)
438     //                                  "----------------\n"
439     std::string diesel_generator_string = "DIESEL GENERATOR\n";
440     diesel_generator_string            += "                \n";
441     diesel_generator_string            += "CAPACITY: 100 kW\n";
```

```
442     diesel_generator_string            += "COST:     ";
443     diesel_generator_string            += std::to_string(DIESEL_GENERATOR_BUILD_COST);
444     diesel_generator_string            += " K\n\n\n";
445     diesel_generator_string            += "BUILD:    [D]   \n";
446 
447     //  3. call general method
448     this->__setUpBuildOption(texture_key, diesel_generator_string);
449 
450     return;
451 }   /* __setUpDieselGeneratorBuildOption() */
```

4.7.3.33 __setUpEnergyStorageSystemBuildOption()

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

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

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

4.7.3.34 __setUpMagnifyingGlassSprite()

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

Helper method to set up and position magnifying glass sprite.

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

4.7.3.35 __setUpNodeSprite()

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

Helper method to set up node sprite.

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

4.7.3.36 __setUpResourceChipSprite()

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

Helper method to set up resource chip sprite.

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

4.7.3.37 __setUpSelectOutlineSprite()

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

Helper method to set up select outline sprite.

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

4.7.3.38 __setUpSolarPVBuildOption()

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

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

Parameters

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() */
```

4.7.3.39 __setUpTidalTurbineBuildOption()

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

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

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

4.7.3.40 __setUpTileExplosionReel()

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

Helper method to set up tile explosion sprite reel.

```
308 {
309     for (int i = 0; i < 4; i++) {
310         for (int j = 0; j < 4; j++) {
311             this->explosion_sprite_reel.push_back(
312                 sf::Sprite(
313                     *(this->assets_manager_ptr->getTexture("tile clear explosion")),
314                     sf::IntRect(j * 64, i * 64, 64, 64)
315                 )
316             );
317 
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() */
```

4.7.3.41 __setUpTileSprite()

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

Helper method to set up tile sprite.

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

4.7.3.42 __setUpWaveEnergyConverterBuildOption()

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

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

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

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

4.7.3.43 __setUpWindTurbineBuildOption()

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

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

Parameters

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() */
```

4.7.3.44 assess()

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

Method to assess the tile's resource.

```
2685 {
2686     this->resource_assessed = true;
2687     this->resource_assessment = true;
2688 
2689     this->assets_manager_ptr->getSound("resource assessment")->play();
2690 
2691     this->__setResourceText();
2692     this->__sendTileStateMessage();
2693 
2694     return;
2695 }   /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2563 {
2564     switch (this->tile_type) {
2565         case (TileType :: FOREST): {
2566             this->tile_decoration_sprite.setTexture(
2567                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2568             );
2569 
2570             break;
2571         }
2572 
2573         case (TileType :: LAKE): {
2574             this->tile_decoration_sprite.setTexture(
2575                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2576             );
2577 
2578             break;
2579         }
2580 
2581         case (TileType :: MOUNTAINS): {
2582             this->tile_decoration_sprite.setTexture(
2583                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2584             );
2585 
2586             break;
2587         }
2588 
2589         case (TileType :: OCEAN): {
2590             this->tile_decoration_sprite.setTexture(
2591                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2592             );
2593 
2594             break;
2595         }
2596 
2597         case (TileType :: PLAINS): {
2598             this->tile_decoration_sprite.setTexture(
2599                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2600             );
2601 
2602             break;
2603         }
2604 
2605         default: {
2606             // do nothing!
2607 
2608             break;
2609         }
2610     }
2611 
2612 
2613     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```
2614         this->tile_decoration_sprite.setOrigin(
2615             this->tile_decoration_sprite.getLocalBounds().width / 2,
2616             this->tile_decoration_sprite.getLocalBounds().height / 2
2617         );
2618 
2619         this->tile_decoration_sprite.setPosition(
2620             this->position_x,
2621             this->position_y
2622         );
2623 
2624         if ((double)rand() / RAND_MAX > 0.5) {
2625             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2626         }
2627     }
2628 
2629     else {
2630         this->tile_decoration_sprite.setOrigin(
2631             this->tile_decoration_sprite.getLocalBounds().width / 2,
2632             this->tile_decoration_sprite.getLocalBounds().height
2633         );
2634 
2635         this->tile_decoration_sprite.setPosition(
2636             this->position_x,
2637             this->position_y + 12
2638         );
2639 
2640         if ((double)rand() / RAND_MAX > 0.5) {
2641             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2642         }
2643     }
2644 
2645     return;
2646 }   /* decorateTile(void) */
```

4.7.3.46 draw()

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

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

```
2821 {
2822     //  1. draw hex
2823     this->render_window_ptr->draw(this->tile_sprite);
2824 
2825     //  2. draw node
2826     if (this->show_node) {
2827         this->render_window_ptr->draw(this->node_sprite);
2828     }
2829 
2830     //  3. draw tile decoration
2831     if (not this->decoration_cleared) {
2832         this->render_window_ptr->draw(this->tile_decoration_sprite);
2833     }
2834 
2835     //  4. draw selection outline
2836     if (this->is_selected) {
2837         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2838 
2839         outline_colour.a =
2840             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2841 
2842         this->select_outline_sprite.setOutlineColor(outline_colour);
2843 
2844         this->render_window_ptr->draw(this->select_outline_sprite);
2845     }
2846 
2847     //  5. draw tile improvement
2848     if (this->has_improvement) {
2849         if (not this->tile_improvement_ptr->just_built) {
2850             this->tile_improvement_ptr->draw();
2851         }
2852     }
2853 
2854     //  6. draw resource
2855     if (this->show_resource) {
2856         this->render_window_ptr->draw(this->resource_chip_sprite);
2857         this->render_window_ptr->draw(this->resource_text);
2858     }
```

```
2859
2860     //  7. draw resource assessment notification
2861     if (this->resource_assessment) {
2862         int alpha = this->magnifying_glass_sprite.getColor().a;
2863
2864         alpha -= 0.05 * FRAMES_PER_SECOND;
2865         if (alpha < 0) {
2866             alpha = 0;
2867             this->resource_assessment = false;
2868         }
2869
2870         this->magnifying_glass_sprite.setColor(
2871             sf::Color(255, 255, 255, alpha)
2872         );
2873
2874         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2875     }
2876
2877     //  8. draw explosion, then settlement placement
2878     if (this->draw_explosion) {
2879         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2880
2881         if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2882             this->explosion_frame++;
2883         }
2884
2885         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2886             this->draw_explosion = false;
2887             this->explosion_frame = 0;
2888         }
2889     }
2890
2891     else if (this->has_improvement) {
2892         if (this->tile_improvement_ptr->just_built) {
2893             this->tile_improvement_ptr->draw();
2894         }
2895     }
2896
2897     //  9. build menu
2898     if (this->build_menu_open) {
2899         this->render_window_ptr->draw(this->build_menu_backing);
2900         this->render_window_ptr->draw(this->build_menu_backing_text);
2901
2902         for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2903             for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2904                 this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2905             }
2906             this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2907         }
2908     }
2909
2910     this->frame++;
2911     return;
2912 }   /* draw() */
```

4.7.3.47 processEvent()

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

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

```
2710 {
2711     //  1. process TileImprovement events
2712     if (
2713         this->is_selected and
2714         this->tile_improvement_ptr != NULL
2715     ) {
2716         this->tile_improvement_ptr->processEvent();
2717     }
2718
2719     //  2. process HexTile events
2720     if (this->event_ptr->type == sf::Event::KeyPressed) {
2721         this->__handleKeyPressEvents();
2722     }
2723
2724     if (this->event_ptr->type == sf::Event::KeyReleased) {
2725         this->__handleKeyReleaseEvents();
2726     }
```

```
2727
2728     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2729         this->__handleMouseButtonEvents();
2730     }
2731
2732     return;
2733 }    /* processEvent() */
```

4.7.3.48 processMessage()

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

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

```
2748 {
2749     //  1. process TileImprovement messages
2750     if (this->tile_improvement_ptr != NULL) {
2751         this->tile_improvement_ptr->processMessage();
2752     }
2753
2754     //  2. process HexTile messages
2755     if (this->is_selected) {
2756         if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2757             Message game_state_message = this->message_hub_ptr->receiveMessage(
2758                 GAME_STATE_CHANNEL
2759             );
2760
2761             if (game_state_message.subject == "game state") {
2762                 this->credits = game_state_message.int_payload["credits"];
2763                 this->game_phase = game_state_message.string_payload["game phase"];
2764
2765                 if (this->tile_improvement_ptr != NULL) {
2766                     this->tile_improvement_ptr->credits = this->credits;
2767                     this->tile_improvement_ptr->game_phase = this->game_phase;
2768
2769                     this->tile_improvement_ptr->month =
2770                         game_state_message.int_payload["month"];
2771
2772                     this->tile_improvement_ptr->demand_MWh =
2773                         game_state_message.int_payload["demand_MWh"];
2774
2775                     this->tile_improvement_ptr->demand_vec_MWh =
2776                         game_state_message.vector_payload["demand_vec_MWh"];
2777
2778                     this->tile_improvement_ptr->update();
2779                 }
2780
2781                 std::cout « "Game state message received by " « this « std::endl;
2782                 this->__sendTileStateMessage();
2783                 this->message_hub_ptr->popMessage(GAME_STATE_CHANNEL);
2784             }
2785         }
2786
2787         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2788             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2789                 TILE_STATE_CHANNEL
2790             );
2791
2792             if (tile_state_message.subject == "state request") {
2793                 this->__sendTileStateMessage();
2794
2795                 std::cout « "Tile state request received by " « this « std::endl;
2796                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2797             }
2798         }
2799
2800         std::cout « "Current credits (HexTile): " « this->credits « " K" «
2801             std::endl;
2802     }
2803
2804     return;
2805 }    /* processMessage() */
```

4.7.3.49 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].

```
2512 {
2513     //  1. check input
2514     if (input_value < 0 or input_value > 1) {
2515         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
2516         error_str += "not in the closed interval [0, 1]";
2517 
2518         #ifdef _WIN32
2519             std::cout « error_str « std::endl;
2520         #endif  /* _WIN32 */
2521 
2522         throw std::runtime_error(error_str);
2523     }
2524 
2525     //  2. convert input value to tile resource
2526     TileResource tile_resource;
2527 
2528     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2529         tile_resource = TileResource :: POOR;
2530     }
2531     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2532         tile_resource = TileResource :: BELOW_AVERAGE;
2533     }
2534     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2535         tile_resource = TileResource :: AVERAGE;
2536     }
2537     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2538         tile_resource = TileResource :: ABOVE_AVERAGE;
2539     }
2540     else {
2541         tile_resource = TileResource :: GOOD;
2542     }
2543 
2544     //  3. call alternate method
2545     this->setTileResource(tile_resource);
2546 
2547     return;
2548 }   /* setTileResource(double) */
```

4.7.3.50 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.

```
2490 {
2491     this->tile_resource = tile_resource;
2492     this->__setResourceText();
2493 
2494     return;
2495 }   /* setTileResource(TileResource) */
```

4.7.3.51 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].

```
2440 {
2441     //  1. check input
2442     if (input_value < 0 or input_value > 1) {
2443         std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
2444         error_str += "not in the closed interval [0, 1]";
2445 
2446         #ifdef _WIN32
2447             std::cout « error_str « std::endl;
2448         #endif  /* _WIN32 */
2449 
2450         throw std::runtime_error(error_str);
2451     }
2452 
2453     //  2. convert input value to tile type
2454     TileType tile_type;
2455 
2456     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2457         tile_type = TileType :: LAKE;
2458     }
2459     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2460         tile_type = TileType :: PLAINS;
2461     }
2462     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2463         tile_type = TileType :: FOREST;
2464     }
2465     else {
2466         tile_type = TileType :: MOUNTAINS;
2467     }
2468 
2469     //  3. call alternate method
2470     this->setTileType(tile_type);
2471 
2472     return;
2473 }   /* setTileType(double) */
```

4.7.3.52 setTileType() [2/2]

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

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

Parameters

tile_type	The type (TileType) to set the tile to.

```
2379 {
2380     this->tile_type = tile_type;
2381 
2382     switch (this->tile_type) {
2383         case (TileType :: FOREST): {
2384             this->tile_sprite.setFillColor(FOREST_GREEN);
2385 
2386             break;
2387         }
2388 
2389         case (TileType :: LAKE): {
```

```
2390             this->tile_sprite.setFillColor(LAKE_BLUE);
2391
2392             break;
2393         }
2394
2395         case (TileType :: MOUNTAINS): {
2396             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2397
2398             break;
2399         }
2400
2401         case (TileType :: OCEAN): {
2402             this->tile_sprite.setFillColor(OCEAN_BLUE);
2403
2404             break;
2405         }
2406
2407         case (TileType :: PLAINS): {
2408             this->tile_sprite.setFillColor(PLAINS_YELLOW);
2409
2410             break;
2411         }
2412
2413         default: {
2414             // do nothing!
2415
2416             break;
2417         }
2418     }
2419
2420     this->__setUpBuildMenu();
2421
2422     return;
2423 }   /* setTileType(TileType) */
```

4.7.3.53 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
2661 {
2662     if (this->show_resource) {
2663         this->show_resource = false;
2664     }
2665     else {
2666         this->show_resource = true;
2667     }
2668
2669     return;
2670 }   /* toggleResourceOverlay() */
```

4.7.4 Member Data Documentation

4.7.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.7.4.2 build_menu_backing

```
sf::RectangleShape HexTile::build_menu_backing
```

A backing for the tile build menu.

4.7.4.3 build_menu_backing_text

```
sf::Text HexTile::build_menu_backing_text
```

A text label for the build menu.

4.7.4.4 build_menu_open

```
bool HexTile::build_menu_open
```

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

4.7.4.5 build_menu_options_text_vec

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

A vector of text for the tile build options.

4.7.4.6 build_menu_options_vec

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

A vector of sprites for illustrating the tile build options.

4.7.4.7 credits

```
int HexTile::credits
```

The current balance of credits.

4.7.4.8 decoration_cleared

```
bool HexTile::decoration_cleared
```

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

4.7.4.9 draw_explosion

```
bool HexTile::draw_explosion
```

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

4.7.4.10 event_ptr

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

A pointer to the event class.

4.7.4.11 explosion_frame

```
size_t HexTile::explosion_frame
```

The current frame of the explosion animation.

4.7.4.12 explosion_sprite_reel

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

A reel of sprites for a tile explosion animation.

4.7.4.13 frame

```
unsigned long long int HexTile::frame
```

The current frame of this object.

4.7.4.14 game_phase

```
std::string HexTile::game_phase
```

The current phase of the game.

4.7.4.15 has_improvement

```
bool HexTile::has_improvement
```

A boolean which indicates if tile has improvement or not.

4.7.4.16 is_selected

```
bool HexTile::is_selected
```

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

4.7.4.17 magnifying_glass_sprite

```
sf::Sprite HexTile::magnifying_glass_sprite
```

A magnifying glass sprite.

4.7.4.18 major_radius

```
double HexTile::major_radius
```

The radius of the smallest bounding circle.

4.7.4.19 message_hub_ptr

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

A pointer to the message hub.

4.7.4.20 minor_radius

```
double HexTile::minor_radius
```

The radius of the largest inscribed circle.

4.7.4.21 node_sprite

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

A circle shape to mark the tile node.

4.7.4.22 position_x

```
double HexTile::position_x
```

The x position of the tile.

4.7.4.23 position_y

```
double HexTile::position_y
```

The y position of the tile.

4.7.4.24 render_window_ptr

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

A pointer to the render window.

4.7.4.25 resource_assessed

```
bool HexTile::resource_assessed
```

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

4.7.4.26 resource_assessment

```
bool HexTile::resource_assessment
```

A boolean which triggers a resource assessment notification.

4.7.4.27 resource_chip_sprite

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

A circle shape which represents a resource chip.

4.7.4.28 resource_text

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

A text representation of the resource.

4.7.4.29 scrap_improvement_frame

```
int HexTile::scrap_improvement_frame
```

A frame for key-hold to confirm scrapping.

4.7.4.30 select_outline_sprite

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

A convex shape which outlines the tile when selected.

4.7.4.31 show_node

```
bool HexTile::show_node
```

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

4.7.4.32 show_resource

```
bool HexTile::show_resource
```

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

4.7.4.33 tile_decoration_sprite

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

A tile decoration sprite.

4.7.4.34 tile_improvement_ptr

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

A pointer to the improvement for this tile.

4.7.4.35 tile_resource

```
TileResource HexTile::tile_resource
```

The renewable resource quality of the tile.

4.7.4.36 tile_sprite

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

A convex shape which represents the tile.

4.7.4.37 tile_type

```
TileType HexTile::tile_type
```

The terrain type of the tile.

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

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

4.8 Message Struct Reference

A structure which defines a standard message format.

`#include <MessageHub.h>`

Public Attributes

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

4.8.1 Detailed Description

A structure which defines a standard message format.

4.8.2 Member Data Documentation

4.8.2.1 bool_payload

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

A boolean payload.

4.8.2.2 channel

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

A string identifying the appropriate channel for this message.

4.8.2.3 double_payload

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

A double payload.

4.8.2.4 int_payload

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

An int payload.

4.8.2.5 string_payload

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

A string payload.

4.8.2.6 subject

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

A string describing the message subject.

4.8.2.7 vector_payload

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

A vector (double) payload.

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

- header/ESC_core/MessageHub.h

4.9 MessageHub Class Reference

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

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

Public Member Functions

- MessageHub (void)

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

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

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

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

  Method for printing message hub state information (mostly for troubleshooting message deadlocks).
- void sendMessage (Message)

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

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

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

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

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

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

  Destructor for the MessageHub class.

Private Attributes

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

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

4.9.1 Detailed Description

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

4.9.2 Constructor & Destructor Documentation

4.9.2.1 MessageHub()

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

Constructor for the MessageHub class.

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

4.9.2.2 ~MessageHub()

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

Destructor for the MessageHub class.

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

4.9.3 Member Function Documentation

4.9.3.1 addChannel()

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

Method to add channel to message map.

Parameters

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.

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

4.9.3.3 clearMessages()

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

Method to clear messages from the MessageHub.

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

4.9.3.4 hasTraffic()

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

Method to determine if there remains any message traffic.

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

4.9.3.5 isEmpty()

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

Method to check if channel is empty.

Parameters

channel	The key for the message channel being checked.

Returns

A boolean indicating whether the channel is empty or not.

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

4.9.3.6 popMessage()

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

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

Parameters

channel	The key for the message channel being popped.

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

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

4.9.3.7 printState()

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

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

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

4.9.3.8 receiveMessage()

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

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

Parameters

channel	The key for the message channel being received from.

Returns

The first message in the given channel.

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

4.9.3.9 removeChannel()

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

Method to remove channel from message map.

Parameters

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() */
```

4.9.3.10 sendMessage()

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

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

Parameters

message	The message to be sent.

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

4.9.4 Member Data Documentation

4.9.4.1 message_map

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

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

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

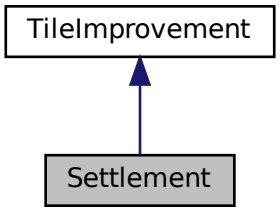
- header/ESC_core/MessageHub.h
- source/ESC_core/MessageHub.cpp

4.10 Settlement Class Reference

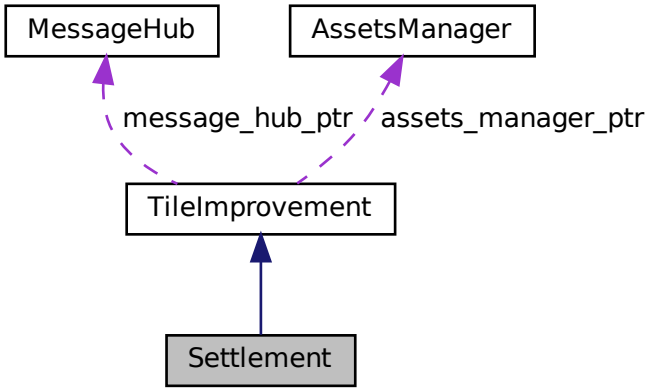
A settlement class (child class of TileImprovement).

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

Inheritance diagram for Settlement:



Collaboration diagram for Settlement:



Public Member Functions

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

  Constructor for the Settlement class.
- void setIsSelected (bool)

  Method to set the is selected attribute.
- std::string getTileOptionsSubstring (void)

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

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

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

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

  Destructor for the Settlement class.

Public Attributes

- bool draw_coin

  Boolean indicating whether or not to draw credits earned coin.
- double smoke_da

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

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

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

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

  A list of smoke sprite (for chimney animation).
- sf::Sprite coin_sprite

  A coin sprite (for credits earned animation).

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

  Helper method to set up and place coin sprite.
- void __handleKeyPressEvents (void)

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.10.1 Detailed Description

A settlement class (child class of TileImprovement).

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Settlement()

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

Constructor for the Settlement class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

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

4.10.2.2 ~Settlement()

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

Destructor for the Settlement class.

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

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

4.10.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.10.3.3 __setUpCoinSprite()

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

Helper method to set up and place coin sprite.

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

4.10.3.4 __setUpTileImprovementSpriteStatic()

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

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

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

4.10.3.5 draw()

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

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

Reimplemented from TileImprovement.

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

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

4.10.3.6 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

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

4.10.3.7 processEvent()

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

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

Reimplemented from TileImprovement.

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

4.10.3.8 processMessage()

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

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

Reimplemented from TileImprovement.

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

4.10.3.9 setIsSelected()

```
void Settlement::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

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

4.10.4 Member Data Documentation

4.10.4.1 coin_sprite

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

A coin sprite (for credits earned animation).

4.10.4.2 draw_coin

```
bool Settlement::draw_coin
```

Boolean indicating whether or not to draw credits earned coin.

4.10.4.3 smoke_da

```
double Settlement::smoke_da
```

The per frame delta in smoke particle alpha value.

4.10.4.4 smoke_dx

```
double Settlement::smoke_dx
```

The per frame delta in smoke particle x position.

4.10.4.5 smoke_dy

```
double Settlement::smoke_dy
```

The per frame delta in smoke particle y position.

4.10.4.6 smoke_prob

```
double Settlement::smoke_prob
```

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

4.10.4.7 smoke_sprite_list

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

A list of smoke sprite (for chimney animation).

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

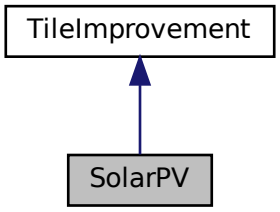
- header/Settlement.h
- source/Settlement.cpp

4.11 SolarPV Class Reference

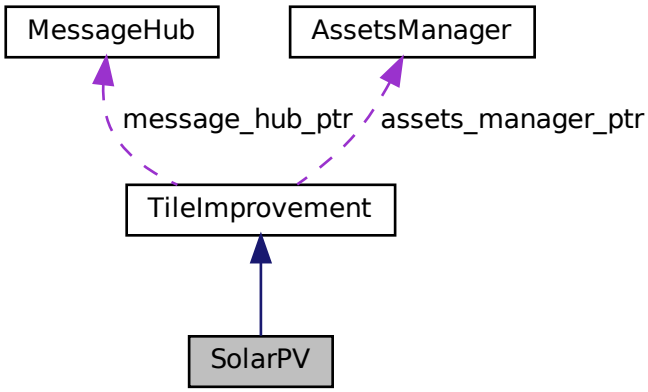
A settlement class (child class of TileImprovement).

`#include <SolarPV.h>`

Inheritance diagram for SolarPV:



Collaboration diagram for SolarPV:



Public Member Functions

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

  Constructor for the SolarPV class.
- std::string getTileOptionsSubstring (void)

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

  Method to set the is selected attribute.
- void advanceTurn (void)

Method to handle turn advance.

- void update (void)

  Method to trigger production and dispatchable updates.
- void processEvent (void)

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

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

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

  Destructor for the SolarPV class.

Public Attributes

- int capacity_kW

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

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

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

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

  The maximum daily production [MWh] of the solar PV array.
- std::vector< double > capacity_factor_vec

  A vector of daily capacity factors for the current month.
- std::vector< double > production_vec_MWh

  A vector of daily production [MWh] for the current month.
- std::vector< double > dispatch_vec_MWh

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

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

  Helper method to draw production menu assets.
- void __upgradePowerCapacity (void)

  Helper method to upgrade power capacity.
- void __computeProductionCosts (void)

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

  Helper method to trigger an equipment breakdown.
- void __computeCapacityFactors (void)

  Helper method to compute capacity factors.
- void __computeProduction (void)

  Helper method to compute production values.
- void __computeDispatch (void)

  Helper method to compute dispatch values.

- void __handleKeyPressEvents (void)

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

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

  Helper method to set up and draw upgrade options.
- void __sendImprovementStateMessage (void)

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.11.1 Detailed Description

A settlement class (child class of TileImprovement).

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SolarPV()

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

Constructor for the SolarPV class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

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

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

4.11.2.2 ~SolarPV()

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

Destructor for the SolarPV class.

```
1051 {
1052     std::cout « "SolarPV at " « this « " destroyed" « std::endl;
1053 
1054     return;
1055 }   /* ~SolarPV() */
```

4.11.3 Member Function Documentation

4.11.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
233 {
```

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

4.11.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

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

4.11.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

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

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

4.11.3.4 __computeProduction()

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

Helper method to compute production values.

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

4.11.3.5 __computeProductionCosts()

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

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

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

4.11.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.11.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

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

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

4.11.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
409 {
410     if (this->just_built) {
411         return;
412     }
413
414     switch (this->event_ptr->key.code) {
415         case (sf::Keyboard::U): {
416             this->__openUpgradeMenu();
417
418             break;
419         }
420
421
422         case (sf::Keyboard::W): {
423             if (this->production_menu_open) {
424                 this->dispatch_MWh++;
425
426                 if (this->dispatch_MWh > this->dispatchable_MWh) {
427                     this->dispatch_MWh = 0;
428                 }
429
430                 this->__computeProductionCosts();
431                 this->assets_manager_ptr->getSound("interface click")->play();
432             }
433
434             else if (this->upgrade_menu_open) {
435                 this->__upgradePowerCapacity();
436             }
437
438             break;
439         }
440
441
442         case (sf::Keyboard::S): {
443             if (this->production_menu_open) {
444                 this->dispatch_MWh--;
445
446                 if (this->dispatch_MWh < 0) {
447                     this->dispatch_MWh = this->dispatchable_MWh;
448                 }
449
450                 this->__computeProductionCosts();
451                 this->assets_manager_ptr->getSound("interface click")->play();
452             }
453
454             break;
455         }
456
457
458         case (sf::Keyboard::D): {
459             if (this->upgrade_menu_open) {
```

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

4.11.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.11.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

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

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

4.11.3.11 __setUpTileImprovementSpriteStatic()

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

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

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

4.11.3.12 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

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

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

4.11.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
854 {
855     //  1. update
856     this->update();
857
858     //  2. send improvement state message
859     this->__sendImprovementStateMessage();
860
861     //  3. handle start/stop
862     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
863         this->is_running = true;
864     }
865
866     else if (this->is_running and (this->dispatch_MWh <= 0)) {
867         this->is_running = false;
868     }
869
870     //  4. handle equipment health
871     if (this->is_running) {
872         this->health--;
873
874         if (this->health <= 0) {
875             this->__breakdown();
876         }
877     }
878
879     return;
880 }   /* advanceTurn() */
```

4.11.3.14 draw()

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

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

Reimplemented from TileImprovement.

```
966 {
967     //  1. if just built, call base method and return
968     if (this->just_built) {
969         TileImprovement :: draw();
970
971         return;
972     }
973
974
975     //  2. handle upgrade effects
976     if (this->just_upgraded) {
977         this->tile_improvement_sprite_static.setColor(
978             sf::Color(
979                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
980                 255,
```

```
981                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
982                 255
983             )
984         );
985 
986         this->tile_improvement_sprite_static.setScale(
987             sf::Vector2f(
988                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
989                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
990             )
991         );
992 
993         this->upgrade_frame++;
994     }
995 
996     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
997         this->tile_improvement_sprite_static.setColor(
998             sf::Color(255,255,255,255)
999         );
1000 
1001          this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
1002 
1003          this->just_upgraded = false;
1004          this->upgrade_frame = 0;
1005     }
1006 
1007 
1008     //  3. draw static sprite
1009     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
1010 
1011 
1012     //  4. draw storage upgrades
1013     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1014         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1015     }
1016 
1017 
1018     //  5. draw production menu
1019     if (this->production_menu_open) {
1020         this->render_window_ptr->draw(this->production_menu_backing);
1021         this->render_window_ptr->draw(this->production_menu_backing_text);
1022 
1023         this->__drawProductionMenu();
1024     }
1025 
1026     //  6. draw upgrade menu
1027     if (this->upgrade_menu_open) {
1028         this->render_window_ptr->draw(this->upgrade_menu_backing);
1029         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1030 
1031         this->__drawUpgradeOptions();
1032     }
1033 
1034     this->frame++;
1035     return;
1036 }   /* draw() */
```

4.11.3.15 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
776 {
777     //                  32 char x 17 line console "--------------------------------\n";
778     std::string options_substring                = "CAPACITY:      ";
779     options_substring                           += std::to_string(this->capacity_kW);
780     options_substring                           += " kW (level ";
```

```
781     options_substring                           += std::to_string(this->upgrade_level);
782     options_substring                           += ")\n";
783
784     options_substring                           += "PRODUCTION:    ";
785     options_substring                           += std::to_string(this->production_MWh);
786     options_substring                           += " MWh\n";
787
788     options_substring                           += "DISPATCHABLE:  ";
789     options_substring                           += std::to_string(this->dispatchable_MWh);
790     options_substring                           += " MWh\n";
791
792     options_substring                           += "HEALTH:      ";
793     options_substring                           += std::to_string(this->health);
794     options_substring                           += "/100";
795
796     if (this->health <= 0) {
797         options_substring                       += " ** BROKEN! **\n";
798     }
799
800     else {
801         options_substring                       += "\n";
802     }
803
804     options_substring                           += "                                \n";
805     options_substring                           += "   **** SOLAR PV OPTIONS ****   \n";
806     options_substring                           += "                                \n";
807     options_substring                           += "     [E]:  OPEN PRODUCTION MENU \n";
808     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
809     options_substring                           += "HOLD [P]:  SCRAP (";
810     options_substring                           += std::to_string(SCRAP_COST);
811     options_substring                           += " K)";
812
813     return options_substring;
814 }   /* getTileOptionsSubstring() */
```

4.11.3.16 processEvent()

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

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

Reimplemented from TileImprovement.

```
917 {
918     TileImprovement :: processEvent();
919
920     if (this->event_ptr->type == sf::Event::KeyPressed) {
921         this->__handleKeyPressEvents();
922     }
923
924     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
925         this->__handleMouseButtonEvents();
926     }
927
928     return;
929 }   /* processEvent() */
```

4.11.3.17 processMessage()

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

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

Reimplemented from TileImprovement.

```
944 {
945     TileImprovement :: processMessage();
946
947     //...
948
949     return;
950 }   /* processMessage() */
```

4.11.3.18 setIsSelected()

```
void SolarPV::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
831 {
832     TileImprovement :: setIsSelected(is_selected);
833 
834     if (this->is_running and this->is_selected) {
835         this->assets_manager_ptr->getSound("solar hum")->play();
836     }
837 
838     return;
839 }   /* setIsSelected() */
```

4.11.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
895 {
896     this->__computeCapacityFactors();
897     this->__computeProduction();
898     this->__computeProductionCosts();
899     this->__computeDispatch();
900 
901     return;
902 }   /* update() */
```

4.11.4 Member Data Documentation

4.11.4.1 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.11.4.2 capacity_kW

```
int SolarPV::capacity_kW
```

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

4.11.4.3 dispatch_MWh

```
int SolarPV::dispatch_MWh
```

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

4.11.4.4 dispatch_vec_MWh

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

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

4.11.4.5 dispatchable_MWh

```
int SolarPV::dispatchable_MWh
```

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

4.11.4.6 max_daily_production_MWh

```
double SolarPV::max_daily_production_MWh
```

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

4.11.4.7 production_MWh

```
int SolarPV::production_MWh
```

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

4.11.4.8 production_vec_MWh

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

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

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

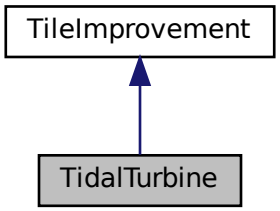
- header/SolarPV.h
- source/SolarPV.cpp

4.12 TidalTurbine Class Reference

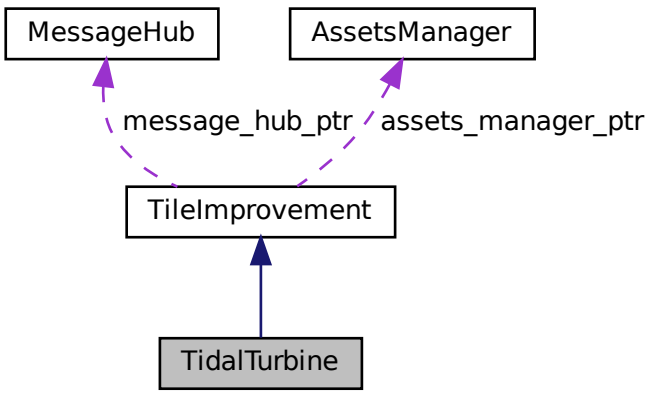
A settlement class (child class of TileImprovement).

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

Inheritance diagram for TidalTurbine:



Collaboration diagram for TidalTurbine:



Public Member Functions

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

  Constructor for the TidalTurbine class.
- std::string getTileOptionsSubstring (void)

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

  Method to set the is selected attribute.
- void advanceTurn (void)

  Method to handle turn advance.
- void update (void)

  Method to trigger production and dispatchable updates.
- void processEvent (void)

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

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

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

  Destructor for the TidalTurbine class.

Public Attributes

- int capacity_kW

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

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

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

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

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

  The rotation rate of the rotor.
- std::vector< double > capacity_factor_vec

  A vector of daily capacity factors for the current month.
- std::vector< double > production_vec_MWh

  A vector of daily production [MWh] for the current month.
- std::vector< double > dispatch_vec_MWh

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

  Helper method to draw production menu assets.
- void __upgradePowerCapacity (void)

  Helper method to upgrade power capacity.
- void __computeProductionCosts (void)

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

  Helper method to trigger an equipment breakdown.
- void __computeCapacityFactors (void)

  Helper method to compute capacity factors.
- void __computeProduction (void)

  Helper method to compute production values.
- void __computeDispatch (void)

  Helper method to compute dispatch values.
- void __handleKeyPressEvents (void)

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

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

  Helper method to set up and draw upgrade options.
- void __sendImprovementStateMessage (void)

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.12.1 Detailed Description

A settlement class (child class of TileImprovement).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TidalTurbine()

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

Constructor for the TidalTurbine class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

```
710    :
711 TileImprovement(
712     position_x,
713     position_y,
714     tile_resource,
715     event_ptr,
716     render_window_ptr,
717     assets_manager_ptr,
718     message_hub_ptr
719 )
720 {
721     //  1. set attributes
722
723     //  1.1. private
724     //...
725
726     //  1.2. public
727     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
728
729     this->is_running = false;
730
731     this->health = 100;
732
733     this->capacity_kW = 100;
734     this->upgrade_level = 1;
735
736     this->storage_kWh = 0;
737     this->storage_level = 0;
738
739     this->production_MWh = 0;
740     this->dispatch_MWh = 0;
741     this->dispatchable_MWh = 0;
742
743     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
744
745     this->capacity_factor_vec.resize(30, 0);
746     this->production_vec_MWh.resize(30, 0);
747     this->dispatch_vec_MWh.resize(30, 0);
748
749     this->tile_improvement_string = "TIDAL TURBINE";
750
751     this->__setUpTileImprovementSpriteAnimated();
752     this->update();
753
754     std::cout « "TidalTurbine constructed at " « this « std::endl;
755
756     return;
757 }   /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

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

Destructor for the TidalTurbine class.

```
1063 {
1064     std::cout « "TidalTurbine at " « this « " destroyed" « std::endl;
1065
1066     return;
1067 }   /* ~TidalTurbine() */
```

4.12.3 Member Function Documentation

4.12.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.12.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

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

4.12.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
324 {
325     double stored_energy_MWh = 0;
326     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
327 
328     double demand_MWh = 0;
329     double production_MWh = 0;
330     double dispatch_MWh = 0;
331     double difference_MWh = 0;
332 
333     double room_MWh = 0;
334 
335     for (int i = 0; i < 30; i++) {
336         demand_MWh = this->demand_vec_MWh[i];
337         production_MWh = this->production_vec_MWh[i];
338 
339         if (production_MWh <= demand_MWh) {
340             this->dispatch_vec_MWh[i] = production_MWh;
341             dispatch_MWh += this->dispatch_vec_MWh[i];
342 
```

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

4.12.3.4 __computeProduction()

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

Helper method to compute production values.

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

4.12.3.5 __computeProductionCosts()

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

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

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

4.12.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.12.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
531 {
532     //  1. draw power capacity upgrade sprite
533     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
534         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
535         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
536 
537         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
538         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
539 
540         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
541         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
542 
543         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
544 
545         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
546         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
547         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
548     }
549 
550     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
551 
552 
553     //  2. draw power capacity upgrade text
554     //                  16 char line = "                \n"
555     std::string power_upgrade_string = "POWER CAPACITY  \n";
556     power_upgrade_string            += "                \n";
557 
558     power_upgrade_string            += "CAPACITY:  ";
559     power_upgrade_string            += std::to_string(this->capacity_kW);
560     power_upgrade_string            += " kW\n";
561 
562     power_upgrade_string            += "LEVEL:     ";
563     power_upgrade_string            += std::to_string(this->upgrade_level);
564     power_upgrade_string            += "\n\n";
565 
566     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
567         power_upgrade_string        += "[W]: + 100 kW (";
568         power_upgrade_string        +=  std::to_string(TIDAL_TURBINE_BUILD_COST);
569         power_upgrade_string        += " K)\n";
570     }
571 
572     else {
573         power_upgrade_string        += " * MAX LEVEL *  \n";
574     }
575 
576     sf::Text power_upgrade_text = sf::Text(
577         power_upgrade_string,
578         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
579         16
580     );
581 
582     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
583     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
584     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
585 
586     this->render_window_ptr->draw(power_upgrade_text);
587 
588 
589     //  3. draw energy capacity (storage) upgrade sprite
590     this->render_window_ptr->draw(this->storage_upgrade_sprite);
591     this->render_window_ptr->draw(this->upgrade_plus_sprite);
592 
593 
594     //  4. draw energy capacity (storage) upgrade text
595     //                   16 char line = "                \n"
596     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
597     energy_upgrade_string            += "                \n";
598 
599     energy_upgrade_string            += "CAPACITY:  ";
600     energy_upgrade_string            += std::to_string(this->storage_level * 200);
601     energy_upgrade_string            += " kWh\n";
602 
603     energy_upgrade_string            += "LEVEL:     ";
604     energy_upgrade_string            += std::to_string(this->storage_level);
605     energy_upgrade_string            += "\n\n";
606 
607     if (this->storage_level < MAX_STORAGE_LEVELS) {
608         energy_upgrade_string        += "[D]: + 200 kWh (";
```

```
609         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
610         energy_upgrade_string        += " K)\n";
611     }
612 
613     else {
614         energy_upgrade_string += " * MAX LEVEL *  \n";
615     }
616 
617     sf::Text energy_upgrade_text = sf::Text(
618         energy_upgrade_string,
619         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
620         16
621     );
622 
623     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
624     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
625     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
626 
627     this->render_window_ptr->draw(energy_upgrade_text);
628 
629     return;
630 }   /* __drawUpgradeOptions() */
```

4.12.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
405 {
406     if (this->just_built) {
407         return;
408     }
409 
410     switch (this->event_ptr->key.code) {
411         case (sf::Keyboard::U): {
412             this->__openUpgradeMenu();
413 
414             break;
415         }
416 
417 
418         case (sf::Keyboard::W): {
419             if (this->production_menu_open) {
420                 this->dispatch_MWh++;
421 
422                 if (this->dispatch_MWh > this->dispatchable_MWh) {
423                     this->dispatch_MWh = 0;
424                 }
425 
426                 this->__computeProductionCosts();
427                 this->assets_manager_ptr->getSound("interface click")->play();
428             }
429 
430             else if (this->upgrade_menu_open) {
431                 this->__upgradePowerCapacity();
432             }
433 
434             break;
435         }
436 
437 
438         case (sf::Keyboard::S): {
439             if (this->production_menu_open) {
440                 this->dispatch_MWh--;
441 
442                 if (this->dispatch_MWh < 0) {
443                     this->dispatch_MWh = this->dispatchable_MWh;
444                 }
445 
446                 this->__computeProductionCosts();
447                 this->assets_manager_ptr->getSound("interface click")->play();
448             }
449 
450             break;
451         }
452 
453 
```

```
454         case (sf::Keyboard::D): {
455             if (this->upgrade_menu_open) {
456                 this->__upgradeStorageCapacity();
457                 this->__computeProduction();
458                 this->__computeDispatch();
459             }
460 
461             break;
462         }
463 
464 
465         default: {
466             // do nothing!
467 
468             break;
469         }
470     }
471 
472     return;
473 }   /* __handleKeyPressEvents() */
```

4.12.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.12.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
645 {
646     Message improvement_state_message;
647 
648     improvement_state_message.channel = GAME_CHANNEL;
649     improvement_state_message.subject = "improvement state";
650 
```

```
651     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
652     improvement_state_message.int_payload["operation_maintenance_cost"] =
653         this->operation_maintenance_cost;
654 
655     this->message_hub_ptr->sendMessage(improvement_state_message);
656 
657     std::cout << "Improvement state message sent by " << this << std::endl;
658 
659     return;
660 }   /* __sendImprovementStateMessage() */
```

4.12.3.11 __setUpTileImprovementSpriteAnimated()

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

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

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

4.12.3.12 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

```
181 {
182     if (this->credits < TIDAL_TURBINE_BUILD_COST) {
183         std::cout << "Cannot upgrade tidal turbine: insufficient credits (need "
184             << TIDAL_TURBINE_BUILD_COST << " K)" << std::endl;
185 
186         this->__sendInsufficientCreditsMessage();
187         return;
188     }
189 
190     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
191         return;
192     }
193 
```

```
194     this->health = 100;
195
196     this->capacity_kW += 100;
197     this->upgrade_level++;
198
199     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
200
201     this->__computeProduction();
202     this->__computeDispatch();
203
204     this->just_upgraded = true;
205
206     this->assets_manager_ptr->getSound("upgrade")->play();
207
208     this->__sendCreditsSpentMessage(TIDAL_TURBINE_BUILD_COST);
209     this->__sendTileStateRequest();
210     this->__sendGameStateRequest();
211
212     return;
213 }   /* __upgradePowerCapacity() */
```

4.12.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
853 {
854     //  1. update
855     this->update();
856
857     //  2. send improvement state message
858     this->__sendImprovementStateMessage();
859
860     //  3. handle start/stop
861     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
862         this->is_running = true;
863     }
864
865     else if (this->is_running and (this->dispatch_MWh <= 0)) {
866         this->is_running = false;
867     }
868
869     //  4. handle equipment health
870     if (this->is_running) {
871         this->health--;
872
873         if (this->health <= 0) {
874             this->__breakdown();
875         }
876     }
877
878     return;
879 }   /* advanceTurn() */
```

4.12.3.14 draw()

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

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

Reimplemented from TileImprovement.

```
965 {
966     //  1. if just built, call base method and return
```

```
967     if (this->just_built) {
968         TileImprovement :: draw();
969 
970         return;
971     }
972 
973 
974     //  2. handle upgrade effects
975     if (this->just_upgraded) {
976         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
977             this->tile_improvement_sprite_animated[i].setColor(
978                 sf::Color(
979                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
980                     255,
981                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
982                     255
983                 )
984             );
985 
986             this->tile_improvement_sprite_animated[i].setScale(
987                 sf::Vector2f(
988                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
989                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
990                 )
991             );
992         }
993 
994         this->upgrade_frame++;
995     }
996 
997     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
998         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
999             this->tile_improvement_sprite_animated[i].setColor(
1000                 sf::Color(255,255,255,255)
1001             );
1002 
1003             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1004         }
1005 
1006         this->just_upgraded = false;
1007         this->upgrade_frame = 0;
1008     }
1009 
1010 
1011     //  3. draw first element of animated sprite
1012     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1013 
1014 
1015     //  4. draw second element of animated sprite
1016     if (this->is_running) {
1017         //...
1018     }
1019 
1020     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1021 
1022 
1023     //  5. draw storage upgrades
1024     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1025         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1026     }
1027 
1028 
1029     //  6. draw production menu
1030     if (this->production_menu_open) {
1031         this->render_window_ptr->draw(this->production_menu_backing);
1032         this->render_window_ptr->draw(this->production_menu_backing_text);
1033 
1034         this->__drawProductionMenu();
1035     }
1036 
1037 
1038     //  7. draw upgrade menu
1039     if (this->upgrade_menu_open) {
1040         this->render_window_ptr->draw(this->upgrade_menu_backing);
1041         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1042 
1043         this->__drawUpgradeOptions();
1044     }
1045 
1046     this->frame++;
1047     return;
1048 }   /* draw() */
```

4.12.3.15 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
774 {
775     //                  32 char x 17 line console "--------------------------------\n";
776     std::string options_substring                = "CAPACITY:      ";
777     options_substring                           += std::to_string(this->capacity_kW);
778     options_substring                           += " kW (level ";
779     options_substring                           += std::to_string(this->upgrade_level);
780     options_substring                           += ")\n";
781 
782     options_substring                           += "PRODUCTION:    ";
783     options_substring                           += std::to_string(this->production_MWh);
784     options_substring                           += " MWh\n";
785 
786     options_substring                           += "DISPATCHABLE:  ";
787     options_substring                           += std::to_string(this->dispatchable_MWh);
788     options_substring                           += " MWh\n";
789 
790     options_substring                           += "HEALTH:      ";
791     options_substring                           += std::to_string(this->health);
792     options_substring                           += "/100";
793 
794     if (this->health <= 0) {
795         options_substring                       += " ** BROKEN! **\n";
796     }
797 
798     else {
799         options_substring                       += "\n";
800     }
801 
802     options_substring                           += "                                \n";
803     options_substring                           += "**** TIDAL TURBINE OPTIONS **** \n";
804     options_substring                           += "                                \n";
805     options_substring                           += "     [E]:  OPEN PRODUCTION MENU \n";
806     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
807     options_substring                           += "HOLD [P]:  SCRAP (";
808     options_substring                           += std::to_string(SCRAP_COST);
809     options_substring                           += " K)";
810 
811     return options_substring;
812 }   /* getTileOptionsSubstring() */
```

4.12.3.16 processEvent()

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

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

Reimplemented from TileImprovement.

```
916 {
917     TileImprovement :: processEvent();
918 
919     if (this->event_ptr->type == sf::Event::KeyPressed) {
920         this->__handleKeyPressEvents();
921     }
922 
923     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
924         this->__handleMouseButtonEvents();
925     }
926 
927     return;
928 }   /* processEvent() */
```

4.12.3.17 processMessage()

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

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

Reimplemented from TileImprovement.

```
943 {
944     TileImprovement :: processMessage();
945
946     //...
947
948     return;
949 }   /* processMessage() */
```

4.12.3.18 setIsSelected()

```
void TidalTurbine::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
829 {
830     TileImprovement :: setIsSelected(is_selected);
831
832     if (this->is_running and this->is_selected) {
833         this->assets_manager_ptr->getSound("water flow")->play();
834     }
835
836     return;
837 }   /* setIsSelected() */
```

4.12.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
894 {
895     this->__computeCapacityFactors();
896     this->__computeProduction();
897     this->__computeProductionCosts();
898     this->__computeDispatch();
899
900     return;
901 }   /* update() */
```

4.12.4 Member Data Documentation

4.12.4.1 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.12.4.2 capacity_kW

```
int TidalTurbine::capacity_kW
```

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

4.12.4.3 dispatch_MWh

```
int TidalTurbine::dispatch_MWh
```

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

4.12.4.4 dispatch_vec_MWh

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

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

4.12.4.5 dispatchable_MWh

```
int TidalTurbine::dispatchable_MWh
```

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

4.12.4.6 max_daily_production_MWh

```
double TidalTurbine::max_daily_production_MWh
```

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

4.12.4.7 production_MWh

```
int TidalTurbine::production_MWh
```

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

4.12.4.8 production_vec_MWh

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

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

4.12.4.9 rotor_drotation

```
double TidalTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

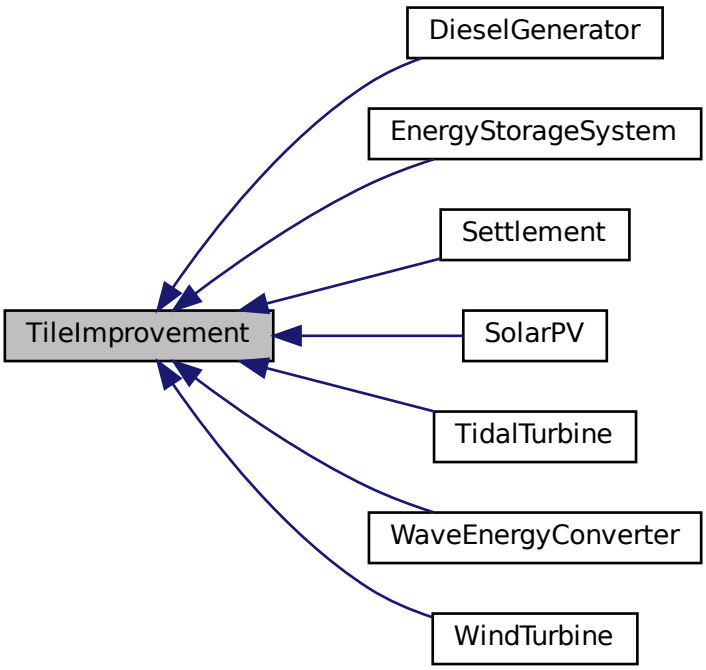
- header/TidalTurbine.h
- source/TidalTurbine.cpp

4.13 TileImprovement Class Reference

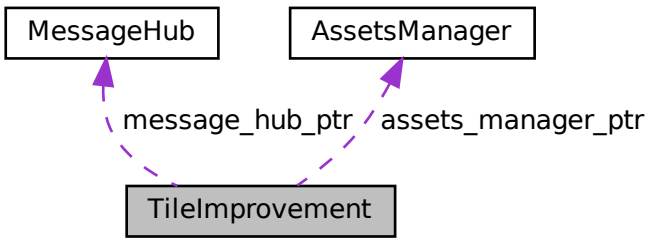
A base class for the tile improvement hierarchy.

`#include <TileImprovement.h>`

Inheritance diagram for TileImprovement:



Collaboration diagram for TileImprovement:



Public Member Functions

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

  Constructor for the TileImprovement class.
- virtual void setIsSelected (bool)

  Method to set the is selected attribute.
- virtual void advanceTurn (void)
- virtual void update (void)
- virtual std::string getTileOptionsSubstring (void)
- virtual void processEvent (void)

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

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

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

  Destructor for the TileImprovement class.

Public Attributes

- TileImprovementType tile_improvement_type

  The type of the tile improvement.
- bool is_running

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

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

  A boolean which indicates that the improvement was just built.
- bool just_upgraded

  A boolean which indicates that the improvement was just upgraded.
- bool production_menu_open

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

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

  A boolean which indicated whether or not improvement is broken.
- unsigned long long int frame

  The current frame of this object.
- int credits

  The current balance of credits.
- int month

  The current month of play.
- int demand_MWh

  The current demand [MWh].
- int health

  The health of the improvement.
- int upgrade_level

  The upgrade level of the improvement.
- int upgrade_frame

  The frame of the upgrade animation.

- int storage_kWh

  The rated energy capacity [kWh] of the storage.
- int storage_level

  The level of storage installed alongside the tile improvement.
- int operation_maintenance_cost

  The operation and maintenance costs for this turn.
- int tile_resource

  The renewable resource quality of the tile.
- double tile_resource_scalar

  A scalar associated with the renewable resource quality.
- double position_x

  The x position of the tile improvement.
- double position_y

  The y position of the tile improvement.
- std::vector< double > demand_vec_MWh

  A vector of daily demands [MWh] for the current month.
- std::string game_phase

  The current phase of the game.
- std::string tile_improvement_string

  A string representation of the tile improvement type.
- sf::Sprite tile_improvement_sprite_static

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

  An animated sprite, for the ContextMenu visual screen.
- sf::RectangleShape production_menu_backing

  A backing for the production menu.
- sf::Text production_menu_backing_text

  Text for the production menu backing.
- sf::RectangleShape upgrade_menu_backing

  A backing for the upgrade menu.
- sf::Text upgrade_menu_backing_text

  Text for the upgrade menu backing.
- sf::Sprite storage_upgrade_sprite

  A sprite for illustrating storage (in upgrade menu).
- std::vector< sf::Sprite > storage_upgrade_sprite_vec

  A vector of sprites for illustrating the storage upgrade level (on tile).
- sf::Sprite upgrade_arrow_sprite

  An upgrade arrow sprite.
- sf::Sprite upgrade_plus_sprite

  An upgrade plus sprite.

Protected Member Functions

- void __setUpProductionMenu (void)

  Helper method to set up and position production menu assets (drawable).
- void __setUpUpgradeMenu (void)

  Helper method to set up and position upgrade menu assets (drawable).
- void __upgradeStorageCapacity (void)

  Helper method to upgrade storage capacity.
- void __handleKeyPressEvents (void)

Helper method to handle key press events.

- void __handleMouseButtonEvents (void)

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

  Helper method to open the production menu.
- void __closeProductionMenu (void)

  Helper method to close the production menu.
- void __breakdown (void)

  Helper method to trigger an equipment breakdown.
- void __openUpgradeMenu (void)

  Helper method to open the upgrade menu.
- void __closeUpgradeMenu (void)

  Helper method to close the build menu.
- void __sendTileStateRequest (void)

  Helper method to format and send a request for the parent HexTile to send a tile state message.
- void __sendGameStateRequest (void)

  Helper method to format and send a game state request (message).
- void __sendCreditsSpentMessage (int)

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

  Helper method to format and send an insufficient credits message.

Protected Attributes

- sf::Event * event_ptr

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

  A pointer to the render window.
- AssetsManager * assets_manager_ptr

  A pointer to the assets manager.
- MessageHub * message_hub_ptr

  A pointer to the message hub.

4.13.1 Detailed Description

A base class for the tile improvement hierarchy.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TileImprovement()

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

Constructor for the TileImprovement class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

```
593 {
594     //  1. set attributes
595 
596     //  1.1. protected
597     this->event_ptr = event_ptr;
598     this->render_window_ptr = render_window_ptr;
599 
600     this->assets_manager_ptr = assets_manager_ptr;
601     this->message_hub_ptr = message_hub_ptr;
602 
603     //  1.2. public
604     this->is_selected = true;
605     this->just_built = true;
606     this->production_menu_open = false;
607     this->upgrade_menu_open = false;
608     this->is_broken = false;
609 
610     this->just_upgraded = false;
611     this->upgrade_frame = 0;
612 
613     this->frame = 0;
614     this->credits = 0;
615     this->month = 1;
616     this->demand_MWh = 0;
617 
618     this->demand_vec_MWh.resize(30, 0);
619 
620     this->operation_maintenance_cost = 0;
621 
622     this->tile_resource = tile_resource;
623 
624     switch (this->tile_resource) {
625         case (0): {
626             this->tile_resource_scalar = 0.7;
627 
628             break;
629         }
630 
631 
632         case (1): {
633             this->tile_resource_scalar = 0.85;
634 
635             break;
636         }
637 
638 
```

```
639         case (2): {
640             this->tile_resource_scalar = 1;
641 
642             break;
643         }
644 
645 
646         case (3): {
647             this->tile_resource_scalar = 1.15;
648 
649             break;
650         }
651 
652 
653         case (4): {
654             this->tile_resource_scalar = 1.3;
655 
656             break;
657         }
658 
659 
660         default: {
661             this->tile_resource_scalar = 1;
662         }
663     }
664 
665     this->position_x = position_x;
666     this->position_y = position_y;
667 
668     this->game_phase = "build settlement";
669 
670     this->__setUpProductionMenu();
671     this->__setUpUpgradeMenu();
672 
673     std::cout << "TileImprovement constructed at " << this << std::endl;
674 
675     return;
676 }   /* TileImprovement() */
```

4.13.2.2 ~TileImprovement()

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

Destructor for the TileImprovement class.

```
904 {
905     std::cout << "TileImprovement at " << this << " destroyed" << std::endl;
906 
907     return;
908 }   /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
377 {
378     this->is_broken = true;
379     this->is_running = false;
380     this->assets_manager_ptr->getSound("breakdown")->play();
381 
382     return;
383 }   /* __breakdown() */
```

4.13.3.2 __closeProductionMenu()

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

Helper method to close the production menu.

```
353 {
354     if (not this->production_menu_open) {
355         return;
356     }
357
358     this->production_menu_open = false;
359     this->assets_manager_ptr->getSound("build menu close")->play();
360
361     return;
362 }   /* __closeProductionMenu() */
```

4.13.3.3 __closeUpgradeMenu()

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

Helper method to close the build menu.

```
426 {
427     if (not this->upgrade_menu_open) {
428         return;
429     }
430
431     this->upgrade_menu_open = false;
432     this->assets_manager_ptr->getSound("build menu close")->play();
433
434     return;
435 }   /* __closeUpgradeMenu() */
```

4.13.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
233 {
234     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
235         return;
236     }
237
238     if (this->just_built) {
239         return;
240     }
241
242     switch (this->event_ptr->key.code) {
243         case (sf::Keyboard::E): {
244             this->__openProductionMenu();
245
246             break;
247         }
248
249
250         default: {
251             // do nothing!
252
253             break;
254         }
255     }
256
257     return;
258 }   /* __handleKeyPressEvents() */
```

4.13.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
273 {
274     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
275         return;
276     }
277
278     if (this->just_built) {
279         return;
280     }
281
282     switch (this->event_ptr->mouseButton.button) {
283         case (sf::Mouse::Left): {
284             //...
285
286             break;
287         }
288
289
290         case (sf::Mouse::Right): {
291             //...
292
293             break;
294         }
295
296
297         default: {
298             // do nothing!
299
300             break;
301         }
302     }
303
304     return;
305 }   /* __handleMouseButtonEvents() */
```

4.13.3.6 __openProductionMenu()

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

Helper method to open the production menu.

```
320 {
321     if (this->is_broken) {
322         this->assets_manager_ptr->getSound("breakdown")->play();
323         return;
324     }
325
326     if (this->production_menu_open) {
327         return;
328     }
329
330     if (this->upgrade_menu_open) {
331         this->__closeUpgradeMenu();
332     }
333
334     this->production_menu_open = true;
335     this->assets_manager_ptr->getSound("build menu open")->play();
336
337     return;
338 }   /* __openProductionMenu() */
```

4.13.3.7 __openUpgradeMenu()

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

Helper method to open the upgrade menu.

```
398 {
399     if (this->upgrade_menu_open) {
400         return;
401     }
402
403     if (this->production_menu_open) {
404         this->__closeProductionMenu();
405     }
406
407     this->upgrade_menu_open = true;
408     this->assets_manager_ptr->getSound("build menu open")->play();
409
410     return;
411 }   /* __openUpgradeMenu() */
```

4.13.3.8 __sendCreditsSpentMessage()

```
void TileImprovement::__sendCreditsSpentMessage (
            int credits_spent )  [protected]
```

Helper method to format and send a credits spent message.

Parameters

credits_spent	The number of credits that were spent.

```
503 {
504     Message credits_spent_message;
505
506     credits_spent_message.channel = GAME_CHANNEL;
507     credits_spent_message.subject = "credits spent";
508
509     credits_spent_message.int_payload["credits spent"] = credits_spent;
510
511     this->message_hub_ptr->sendMessage(credits_spent_message);
512
513     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
514         << std::endl;
515     return;
516 }   /* __sendCreditsSpentMessage() */
```

4.13.3.9 __sendGameStateRequest()

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

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

```
476 {
477     Message game_state_request;
478
479     game_state_request.channel = GAME_CHANNEL;
480     game_state_request.subject = "state request";
481
482     this->message_hub_ptr->sendMessage(game_state_request);
483
484     std::cout << "Game state request message sent by " << this << std::endl;
485     return;
486 }   /* __sendGameStateRequest() */
```

4.13.3.10 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
531 {
532     Message insufficient_credits_message;
533 
534     insufficient_credits_message.channel = GAME_CHANNEL;
535     insufficient_credits_message.subject = "insufficient credits";
536 
537     this->message_hub_ptr->sendMessage(insufficient_credits_message);
538 
539     std::cout << "Insufficient credits message sent by " << this << std::endl;
540 
541     return;
542 }   /* __sendInsufficientCreditsMessage() */
```

4.13.3.11 __sendTileStateRequest()

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

Helper method to format and send a request for the parent HexTile to send a tile state message.

```
451 {
452     Message tile_state_request;
453 
454     tile_state_request.channel = TILE_STATE_CHANNEL;
455     tile_state_request.subject = "state request";
456 
457     this->message_hub_ptr->sendMessage(tile_state_request);
458 
459     std::cout << "Tile state request sent by " << this << std::endl;
460     return;
461 }   /* __sendTileStateRequest() */
```

4.13.3.12 __setUpProductionMenu()

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

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

```
68 {
69     //  1. set up and place production menu backing and text
70     this->production_menu_backing.setSize(sf::Vector2f(400, 256));
71     this->production_menu_backing.setOrigin(200, 128);
72     this->production_menu_backing.setPosition(400, 400);
73     this->production_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
74     this->production_menu_backing.setOutlineColor(MENU_FRAME_GREY);
75     this->production_menu_backing.setOutlineThickness(4);
76 
77     this->production_menu_backing_text.setString("**** PRODUCTION MENU ****");
78     this->production_menu_backing_text.setFont(
79         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
80     );
81     this->production_menu_backing_text.setCharacterSize(16);
82     this->production_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
83     this->production_menu_backing_text.setOrigin(
84         this->production_menu_backing_text.getLocalBounds().width / 2, 0
85     );
86     this->production_menu_backing_text.setPosition(400, 400 - 128 + 4);
87 
88     return;
89 }   /* __setUpProductionMenu() */
```

4.13.3.13 __setUpUpgradeMenu()

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

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

```
104 {
105     //  1. set up and place upgrade menu backing and text
106     this->upgrade_menu_backing.setSize(sf::Vector2f(400, 256));
107     this->upgrade_menu_backing.setOrigin(200, 128);
108     this->upgrade_menu_backing.setPosition(400, 400);
109     this->upgrade_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
110     this->upgrade_menu_backing.setOutlineColor(MENU_FRAME_GREY);
111     this->upgrade_menu_backing.setOutlineThickness(4);
112 
113     this->upgrade_menu_backing_text.setString("**** UPGRADE MENU ****");
114     this->upgrade_menu_backing_text.setFont(
115         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
116     );
117     this->upgrade_menu_backing_text.setCharacterSize(16);
118     this->upgrade_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
119     this->upgrade_menu_backing_text.setOrigin(
120         this->upgrade_menu_backing_text.getLocalBounds().width / 2, 0
121     );
122     this->upgrade_menu_backing_text.setPosition(400, 400 - 128 + 4);
123 
124 
125     //  2. set up and place storage upgrade sprite (with upgrade plus)
126     this->storage_upgrade_sprite = sf::Sprite(
127         *(this->assets_manager_ptr->getTexture("energy storage system"))
128     );
129 
130     this->storage_upgrade_sprite.setOrigin(
131         this->storage_upgrade_sprite.getLocalBounds().width / 2,
132         this->storage_upgrade_sprite.getLocalBounds().height
133     );
134 
135     this->storage_upgrade_sprite.setPosition(400 + 100, 400 - 32);
136 
137     this->upgrade_plus_sprite = sf::Sprite(
138         *(this->assets_manager_ptr->getTexture("upgrade plus"))
139     );
140 
141     this->upgrade_plus_sprite.setOrigin(
142         this->upgrade_plus_sprite.getLocalBounds().width / 2,
143         this->upgrade_plus_sprite.getLocalBounds().height / 2
144     );
145 
146     this->upgrade_plus_sprite.setPosition(400 + 130, 400 - 64);
147 
148 
149     //  3. set up and place upgrade arrow sprite
150     this->upgrade_arrow_sprite = sf::Sprite(
151         *(this->assets_manager_ptr->getTexture("upgrade arrow"))
152     );
153 
154     this->upgrade_arrow_sprite.setOrigin(
155         this->upgrade_arrow_sprite.getLocalBounds().width / 2,
156         this->upgrade_arrow_sprite.getLocalBounds().height / 2
157     );
158 
159     this->upgrade_arrow_sprite.setPosition(400 - 64, 400 - 64);
160 
161 
162     return;
163 }   /* __setUpUpgradeMenu() */
```

4.13.3.14 __upgradeStorageCapacity()

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

Helper method to upgrade storage capacity.

```
178 {
179     if (this->credits < ENERGY_STORAGE_SYSTEM_BUILD_COST) {
```

```
180         std::cout « "Cannot add energy storage: insufficient credits (need "
181             « ENERGY_STORAGE_SYSTEM_BUILD_COST « " K)" « std::endl;
182 
183         this->__sendInsufficientCreditsMessage();
184         return;
185     }
186 
187     if (this->storage_level >= MAX_STORAGE_LEVELS) {
188         return;
189     }
190 
191     this->storage_level++;
192     this->storage_kWh += 200;
193 
194     this->storage_upgrade_sprite_vec.push_back(
195         sf::Sprite(
196             *(this->assets_manager_ptr->getTexture("storage level"))
197         )
198     );
199 
200     this->storage_upgrade_sprite_vec.back().setOrigin(
201         this->storage_upgrade_sprite_vec.back().getLocalBounds().width / 2,
202         this->storage_upgrade_sprite_vec.back().getLocalBounds().height
203     );
204 
205     this->storage_upgrade_sprite_vec.back().setPosition(
206         this->position_x + 18,
207         this->position_y + 25 - 7 * this->storage_upgrade_sprite_vec.size()
208     );
209 
210     this->just_upgraded = true;
211 
212     this->assets_manager_ptr->getSound("upgrade")->play();
213 
214     this->__sendCreditsSpentMessage(ENERGY_STORAGE_SYSTEM_BUILD_COST);
215     this->__sendTileStateRequest();
216 
217     return;
218 }   /* __upgradeStorageCapacity() */
```

4.13.3.15 advanceTurn()

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, and DieselGenerator.

```
184 {return;}
```

4.13.3.16 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.

```
775 {
776     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
777         int alpha = this->tile_improvement_sprite_static.getColor().a;
778 
779         alpha += 0.08 * FRAMES_PER_SECOND;
780 
781         this->tile_improvement_sprite_static.setColor(
782             sf::Color(255, 255, 255, alpha)
783         );
784 
785         this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);
```

```
786
787         if (
788             (alpha >= 255) or
789             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
790         ) {
791             this->tile_improvement_sprite_static.setColor(
792                 sf::Color(255, 255, 255, 255)
793             );
794
795             this->tile_improvement_sprite_static.setPosition(
796                 this->position_x,
797                 this->position_y + 12
798             );
799
800             this->just_built = false;
801             this->assets_manager_ptr->getSound("place improvement")->play();
802         }
803
804         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
805     }
806
807
808     else {
809         int alpha = 0;
810
811         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
812             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
813
814             alpha += 0.08 * FRAMES_PER_SECOND;
815
816             this->tile_improvement_sprite_animated[i].setColor(
817                 sf::Color(255, 255, 255, alpha)
818             );
819
820             this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
821
822             if (
823                 (alpha >= 255) or
824                 (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
825             ) {
826                 this->tile_improvement_sprite_animated[i].setColor(
827                     sf::Color(255, 255, 255, 255)
828                 );
829
830                 this->tile_improvement_sprite_animated[i].setPosition(
831                     this->position_x,
832                     this->position_y + 12
833                 );
834             }
835
836             this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
837         }
838
839         if (
840             (alpha >= 255) or
841             (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
842         ) {
843             this->just_built = false;
844             this->assets_manager_ptr->getSound("place improvement")->play();
845
846             switch (this->tile_improvement_type) {
847                 case (TileImprovementType :: WIND_TURBINE): {
848                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
849                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
850                         this->tile_improvement_sprite_animated[i].move(0, -32);
851                     }
852
853                     break;
854                 }
855
856
857                 case (TileImprovementType :: TIDAL_TURBINE): {
858                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
859                         this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
860                         this->tile_improvement_sprite_animated[i].move(0, -19);
861                     }
862
863                     break;
864                 }
865
866
867                 case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
868                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
869                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
870                         this->tile_improvement_sprite_animated[i].move(0, -32);
871                     }
872
```

```
873                     break;
874                 }
875
876
877                 default: {
878                     // do nothing!
879
880                     break;
881                 }
882             }
883         }
884     }
885
886
887     this->frame++;
888     return;
889 }   /* draw() */
```

4.13.3.17 getTileOptionsSubstring()

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
188 {return "";}
```

4.13.3.18 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.

```
720 {
721     if (this->event_ptr->type == sf::Event::KeyPressed) {
722         this->__handleKeyPressEvents();
723     }
724
725     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
726         this->__handleMouseButtonEvents();
727     }
728
729     return;
730 }   /* processEvent() */
```

4.13.3.19 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.

```
745 {
746     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
747         Message game_state_message = this->message_hub_ptr->receiveMessage(
748             GAME_STATE_CHANNEL
749         );
750 
751         if (game_state_message.subject == "turn advance") {
752             this->advanceTurn();
753 
754             std::cout « "Turn advance message read and passed by " « this « std::endl;
755         }
756     }
757 
758     return;
759 }   /* processMessage() */
```

4.13.3.20 setIsSelected()

```
void TileImprovement::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
693 {
694     this->is_selected = is_selected;
695 
696     if ((not is_selected) and this->production_menu_open) {
697         this->__closeProductionMenu();
698     }
699 
700     if ((not is_selected) and this->upgrade_menu_open) {
701         this->__closeUpgradeMenu();
702     }
703 
704     return;
705 }   /* setIsSelected() */
```

4.13.3.21 update()

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, and SolarPV.

```
186 {return;}
```

4.13.4 Member Data Documentation

4.13.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.13.4.2 credits

```
int TileImprovement::credits
```

The current balance of credits.

4.13.4.3 demand_MWh

```
int TileImprovement::demand_MWh
```

The current demand [MWh].

4.13.4.4 demand_vec_MWh

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

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

4.13.4.5 event_ptr

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

A pointer to the event class.

4.13.4.6 frame

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

The current frame of this object.

4.13.4.7 game_phase

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

The current phase of the game.

4.13.4.8 health

```
int TileImprovement::health
```

The health of the improvement.

4.13.4.9 is_broken

```
bool TileImprovement::is_broken
```

A boolean which indicated whether or not improvement is broken.

4.13.4.10 is_running

```
bool TileImprovement::is_running
```

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

4.13.4.11 is_selected

```
bool TileImprovement::is_selected
```

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

4.13.4.12 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.13 just_upgraded

```
bool TileImprovement::just_upgraded
```

A boolean which indicates that the improvement was just upgraded.

4.13.4.14 message_hub_ptr

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

A pointer to the message hub.

4.13.4.15 month

```
int TileImprovement::month
```

The current month of play.

4.13.4.16 operation_maintenance_cost

```
int TileImprovement::operation_maintenance_cost
```

The operation and maintenance costs for this turn.

4.13.4.17 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.18 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.19 production_menu_backing

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

A backing for the production menu.

4.13.4.20 production_menu_backing_text

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

Text for the production menu backing.

4.13.4.21 production_menu_open

```
bool TileImprovement::production_menu_open
```

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

4.13.4.22 render_window_ptr

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

A pointer to the render window.

4.13.4.23 storage_kWh

```
int TileImprovement::storage_kWh
```

The rated energy capacity [kWh] of the storage.

4.13.4.24 storage_level

```
int TileImprovement::storage_level
```

The level of storage installed alongside the tile improvement.

4.13.4.25 storage_upgrade_sprite

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

A sprite for illustrating storage (in upgrade menu).

4.13.4.26 storage_upgrade_sprite_vec

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

A vector of sprites for illustrating the storage upgrade level (on tile).

4.13.4.27 tile_improvement_sprite_animated

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

An animated sprite, for the ContextMenu visual screen.

4.13.4.28 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.29 tile_improvement_string

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

A string representation of the tile improvement type.

4.13.4.30 tile_improvement_type

```
TileImprovementType TileImprovement::tile_improvement_type
```

The type of the tile improvement.

4.13.4.31 tile_resource

```
int TileImprovement::tile_resource
```

The renewable resource quality of the tile.

4.13.4.32 tile_resource_scalar

```
double TileImprovement::tile_resource_scalar
```

A scalar associated with the renewable resource quality.

4.13.4.33 upgrade_arrow_sprite

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

An upgrade arrow sprite.

4.13.4.34 upgrade_frame

```
int TileImprovement::upgrade_frame
```

The frame of the upgrade animation.

4.13.4.35 upgrade_level

```
int TileImprovement::upgrade_level
```

The upgrade level of the improvement.

4.13.4.36 upgrade_menu_backing

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

A backing for the upgrade menu.

4.13.4.37 upgrade_menu_backing_text

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

Text for the upgrade menu backing.

4.13.4.38 upgrade_menu_open

```
bool TileImprovement::upgrade_menu_open
```

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

4.13.4.39 upgrade_plus_sprite

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

An upgrade plus sprite.

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

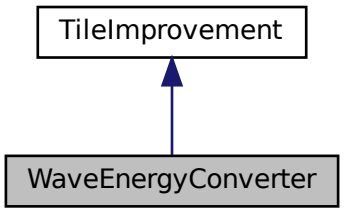
- header/TileImprovement.h
- source/TileImprovement.cpp

4.14 WaveEnergyConverter Class Reference

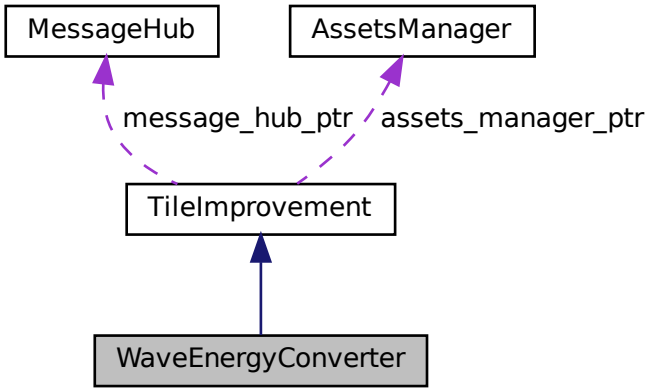
A settlement class (child class of TileImprovement).

`#include <WaveEnergyConverter.h>`

Inheritance diagram for WaveEnergyConverter:



Collaboration diagram for WaveEnergyConverter:



Public Member Functions

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

  Constructor for the WaveEnergyConverter class.
- std::string getTileOptionsSubstring (void)

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

  Method to set the is selected attribute.

- void advanceTurn (void)

  Method to handle turn advance.
- void update (void)

  Method to trigger production and dispatchable updates.
- void processEvent (void)

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

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

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

  Destructor for the WaveEnergyConverter class.

Public Attributes

- int capacity_kW

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

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

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

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

  The maximum daily production [MWh] of the solar PV array.
- std::vector< double > capacity_factor_vec

  A vector of daily capacity factors for the current month.
- std::vector< double > production_vec_MWh

  A vector of daily production [MWh] for the current month.
- std::vector< double > dispatch_vec_MWh

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

  Helper method to draw production menu assets.
- void __upgradePowerCapacity (void)

  Helper method to upgrade power capacity.
- void __computeProductionCosts (void)

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

  Helper method to trigger an equipment breakdown.
- void __computeCapacityFactors (void)

  Helper method to compute capacity factors.
- void __computeProduction (void)

  Helper method to compute production values.
- void __computeDispatch (void)

Helper method to compute dispatch values.

- void __handleKeyPressEvents (void)

  Helper method to handle key press events.

- void __handleMouseButtonEvents (void)

  Helper method to handle mouse button events.

- void __drawUpgradeOptions (void)

  Helper method to set up and draw upgrade options.

- void __sendImprovementStateMessage (void)

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.14.1 Detailed Description

A settlement class (child class of TileImprovement).

4.14.2 Constructor & Destructor Documentation

4.14.2.1 WaveEnergyConverter()

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

Constructor for the WaveEnergyConverter class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

```
730   :
731 TileImprovement(
732     position_x,
733     position_y,
```

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

4.14.2.2 ~WaveEnergyConverter()

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

Destructor for the WaveEnergyConverter class.

```
1086 {
1087     std::cout « "WaveEnergyConverter at " « this « " destroyed" « std::endl;
1088 
1089     return;
1090 }   /* ~WaveEnergyConverter() */
```

4.14.3 Member Function Documentation

4.14.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.14.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

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

4.14.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
345 {
346     double stored_energy_MWh = 0;
347     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
348 
349     double demand_MWh = 0;
350     double production_MWh = 0;
351     double dispatch_MWh = 0;
352     double difference_MWh = 0;
353 
354     double room_MWh = 0;
355 
356     for (int i = 0; i < 30; i++) {
357         demand_MWh = this->demand_vec_MWh[i];
358         production_MWh = this->production_vec_MWh[i];
359 
360         if (production_MWh <= demand_MWh) {
```

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

4.14.3.4 __computeProduction()

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

Helper method to compute production values.

```
317 {
318     double production_MWh = 0;
319 
320     for (int i = 0; i < 30; i++) {
321         this->production_vec_MWh[i] =
322             this->max_daily_production_MWh * this->capacity_factor_vec[i];
323 
324         production_MWh += this->production_vec_MWh[i];
325     }
326 
327     this->production_MWh = round(production_MWh);
328 
329     return;
330 }   /* __computeProduction() */
```

4.14.3.5 __computeProductionCosts()

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

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

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

4.14.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.14.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
551 {
552     //  1. draw power capacity upgrade sprite
553     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
554         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
555         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
556 
557         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
558         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
559 
560         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
561         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
562 
563         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
564 
565         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
566         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
567         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
568     }
569 
570     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
571 
572 
573     //  2. draw power capacity upgrade text
574     //                  16 char line = "                \n"
575     std::string power_upgrade_string = "POWER CAPACITY  \n";
576     power_upgrade_string            += "                \n";
577 
578     power_upgrade_string            += "CAPACITY:  ";
579     power_upgrade_string            += std::to_string(this->capacity_kW);
580     power_upgrade_string            += " kW\n";
581 
582     power_upgrade_string            += "LEVEL:     ";
583     power_upgrade_string            += std::to_string(this->upgrade_level);
584     power_upgrade_string            += "\n\n";
585 
586     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
587         power_upgrade_string        += "[W]: + 100 kW (";
588         power_upgrade_string        +=  std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
589         power_upgrade_string        += " K)\n";
590     }
591 
592     else {
593         power_upgrade_string        += " * MAX LEVEL *  \n";
594     }
595 
596     sf::Text power_upgrade_text = sf::Text(
597         power_upgrade_string,
598         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
599         16
600     );
601 
602     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
603     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
604     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
605 
606     this->render_window_ptr->draw(power_upgrade_text);
607 
608 
609     //  3. draw energy capacity (storage) upgrade sprite
610     this->render_window_ptr->draw(this->storage_upgrade_sprite);
611     this->render_window_ptr->draw(this->upgrade_plus_sprite);
612 
613 
614     //  4. draw energy capacity (storage) upgrade text
615     //                   16 char line = "                \n"
616     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
617     energy_upgrade_string            += "                \n";
618 
619     energy_upgrade_string            += "CAPACITY:  ";
620     energy_upgrade_string            += std::to_string(this->storage_level * 200);
621     energy_upgrade_string            += " kWh\n";
622 
623     energy_upgrade_string            += "LEVEL:     ";
624     energy_upgrade_string            += std::to_string(this->storage_level);
625     energy_upgrade_string            += "\n\n";
626 
627     if (this->storage_level < MAX_STORAGE_LEVELS) {
628         energy_upgrade_string        += "[D]: + 200 kWh (";
```

```
629         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
630         energy_upgrade_string        += " K)\n";
631     }
632 
633     else {
634         energy_upgrade_string += " * MAX LEVEL *  \n";
635     }
636 
637     sf::Text energy_upgrade_text = sf::Text(
638         energy_upgrade_string,
639         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
640         16
641     );
642 
643     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
644     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
645     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
646 
647     this->render_window_ptr->draw(energy_upgrade_text);
648 
649     return;
650 }   /* __drawUpgradeOptions() */
```

4.14.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
426 {
427     if (this->just_built) {
428         return;
429     }
430 
431     switch (this->event_ptr->key.code) {
432         case (sf::Keyboard::U): {
433             this->__openUpgradeMenu();
434 
435             break;
436         }
437 
438 
439         case (sf::Keyboard::W): {
440             if (this->production_menu_open) {
441                 this->dispatch_MWh++;
442 
443                 if (this->dispatch_MWh > this->dispatchable_MWh) {
444                     this->dispatch_MWh = 0;
445                 }
446 
447                 this->__computeProductionCosts();
448                 this->assets_manager_ptr->getSound("interface click")->play();
449             }
450 
451             else if (this->upgrade_menu_open) {
452                 this->__upgradePowerCapacity();
453             }
454 
455             break;
456         }
457 
458 
459         case (sf::Keyboard::S): {
460             if (this->production_menu_open) {
461                 this->dispatch_MWh--;
462 
463                 if (this->dispatch_MWh < 0) {
464                     this->dispatch_MWh = this->dispatchable_MWh;
465                 }
466 
467                 this->__computeProductionCosts();
468                 this->assets_manager_ptr->getSound("interface click")->play();
469             }
470 
471             break;
472         }
473 
474 
```

```
475         case (sf::Keyboard::D): {
476             if (this->upgrade_menu_open) {
477                 this->__upgradeStorageCapacity();
478                 this->__computeProduction();
479                 this->__computeDispatch();
480             }
481 
482             break;
483         }
484 
485 
486         default: {
487             // do nothing!
488 
489             break;
490         }
491     }
492 
493     return;
494 }   /* __handleKeyPressEvents() */
```

4.14.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.14.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
665 {
666     Message improvement_state_message;
667 
668     improvement_state_message.channel = GAME_CHANNEL;
669     improvement_state_message.subject = "improvement state";
670 
671     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
```

```
672     improvement_state_message.int_payload["operation_maintenance_cost"] =
673         this->operation_maintenance_cost;
674
675     this->message_hub_ptr->sendMessage(improvement_state_message);
676
677     std::cout « "Improvement state message sent by " « this « std::endl;
678
679     return;
680 }   /* __sendImprovementStateMessage() */
```

4.14.3.11 __setUpTileImprovementSpriteAnimated()

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

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

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

4.14.3.12 __upgradePowerCapacity()

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

Helper method to upgrade power capacity.

```
181 {
182     if (this->credits < WAVE_ENERGY_CONVERTER_BUILD_COST) {
183         std::cout « "Cannot upgrade wave energy converter: insufficient credits (need "
184             « WAVE_ENERGY_CONVERTER_BUILD_COST « " K)" « std::endl;
185
186         this->__sendInsufficientCreditsMessage();
187         return;
188     }
189
190     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
191         return;
192     }
193
194     this->health = 100;
```

```
195
196     this->capacity_kW += 100;
197     this->upgrade_level++;
198
199     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
200
201     this->__computeProduction();
202     this->__computeDispatch();
203
204     this->just_upgraded = true;
205
206     this->assets_manager_ptr->getSound("upgrade")->play();
207
208     this->__sendCreditsSpentMessage(WAVE_ENERGY_CONVERTER_BUILD_COST);
209     this->__sendTileStateRequest();
210     this->__sendGameStateRequest();
211
212     return;
213 }   /* __upgradePowerCapacity() */
```

4.14.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

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

4.14.3.14 draw()

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

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

Reimplemented from TileImprovement.

```
984 {
985     //  1. if just built, call base method and return
986     if (this->just_built) {
```

```
987         TileImprovement :: draw();
988
989         return;
990     }
991
992
993     //  2. handle upgrade effects
994     if (this->just_upgraded) {
995         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
996             this->tile_improvement_sprite_animated[i].setColor(
997                 sf::Color(
998                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
999                     255,
1000                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1001                     255
1002                 )
1003             );
1004
1005             this->tile_improvement_sprite_animated[i].setScale(
1006                 sf::Vector2f(
1007                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1008                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1009                 )
1010             );
1011         }
1012
1013         this->upgrade_frame++;
1014     }
1015
1016     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1017         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1018             this->tile_improvement_sprite_animated[i].setColor(
1019                 sf::Color(255,255,255,255)
1020             );
1021
1022             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1023         }
1024
1025         this->just_upgraded = false;
1026         this->upgrade_frame = 0;
1027     }
1028
1029
1030     //  3. draw first element of animated sprite
1031     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1032
1033
1034     //  4. draw second element of animated sprite
1035     if (this->is_running) {
1036         //...
1037     }
1038
1039     else {
1040         //...
1041     }
1042
1043     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1044
1045
1046     //  5. draw storage upgrades
1047     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1048         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1049     }
1050
1051
1052     //  6. draw production menu
1053     if (this->production_menu_open) {
1054         this->render_window_ptr->draw(this->production_menu_backing);
1055         this->render_window_ptr->draw(this->production_menu_backing_text);
1056
1057         this->__drawProductionMenu();
1058     }
1059
1060
1061     //  7. draw upgrade menu
1062     if (this->upgrade_menu_open) {
1063         this->render_window_ptr->draw(this->upgrade_menu_backing);
1064         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1065
1066         this->__drawUpgradeOptions();
1067     }
1068
1069     this->frame++;
1070     return;
1071 }   /* draw() */
```

4.14.3.15 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

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

4.14.3.16 processEvent()

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

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

Reimplemented from TileImprovement.

```
935 {
936     TileImprovement :: processEvent();
937 
938     if (this->event_ptr->type == sf::Event::KeyPressed) {
939         this->__handleKeyPressEvents();
940     }
941 
942     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
943         this->__handleMouseButtonEvents();
944     }
945 
946     return;
947 }   /* processEvent() */
```

4.14.3.17 processMessage()

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

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

Reimplemented from TileImprovement.

```
962 {
963     TileImprovement :: processMessage();
964
965     //...
966
967     return;
968 }   /* processMessage() */
```

4.14.3.18 setIsSelected()

```
void WaveEnergyConverter::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
849 {
850     TileImprovement :: setIsSelected(is_selected);
851
852     if (this->is_running and this->is_selected) {
853         this->assets_manager_ptr->getSound("ocean waves")->play();
854     }
855
856     return;
857 }   /* setIsSelected() */
```

4.14.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
913 {
914     this->__computeCapacityFactors();
915     this->__computeProduction();
916     this->__computeProductionCosts();
917     this->__computeDispatch();
918
919     return;
920 }   /* update() */
```

4.14.4 Member Data Documentation

4.14.4.1 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.14.4.2 capacity_kW

```
int WaveEnergyConverter::capacity_kW
```

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

4.14.4.3 dispatch_MWh

```
int WaveEnergyConverter::dispatch_MWh
```

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

4.14.4.4 dispatch_vec_MWh

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

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

4.14.4.5 dispatchable_MWh

```
int WaveEnergyConverter::dispatchable_MWh
```

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

4.14.4.6 max_daily_production_MWh

```
double WaveEnergyConverter::max_daily_production_MWh
```

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

4.14.4.7 production_MWh

```
int WaveEnergyConverter::production_MWh
```

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

4.14.4.8 production_vec_MWh

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

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

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

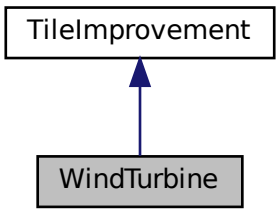
- header/WaveEnergyConverter.h
- source/WaveEnergyConverter.cpp

4.15 WindTurbine Class Reference

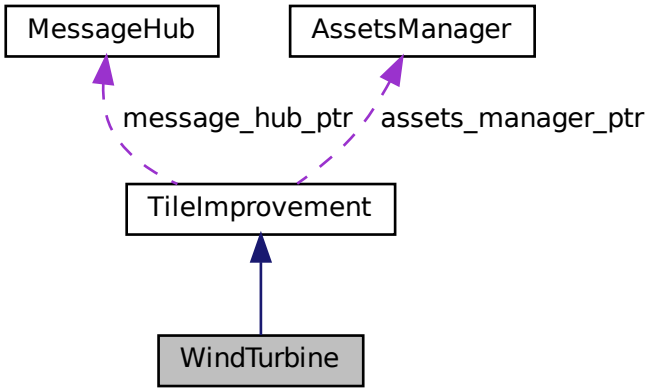
A settlement class (child class of TileImprovement).

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

Inheritance diagram for WindTurbine:



Collaboration diagram for WindTurbine:



Public Member Functions

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

  Constructor for the WindTurbine class.
- std::string getTileOptionsSubstring (void)

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

  Method to set the is selected attribute.
- void advanceTurn (void)

  Method to handle turn advance.
- void update (void)

  Method to trigger production and dispatchable updates.
- void processEvent (void)

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

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

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

  Destructor for the WindTurbine class.

Public Attributes

- int capacity_kW

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

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

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

- int dispatchable_MWh

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

  The maximum daily production [MWh] of the solar PV array.
- std::vector< double > capacity_factor_vec

  A vector of daily capacity factors for the current month.
- std::vector< double > production_vec_MWh

  A vector of daily production [MWh] for the current month.
- std::vector< double > dispatch_vec_MWh

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

  Helper method to draw production menu assets.
- void __upgradePowerCapacity (void)

  Helper method to upgrade the power capacity.
- void __computeProductionCosts (void)

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

  Helper method to trigger an equipment breakdown.
- void __computeCapacityFactors (void)

  Helper method to compute capacity factors.
- void __computeProduction (void)

  Helper method to compute production values.
- void __computeDispatch (void)

  Helper method to compute dispatch values.
- void __handleKeyPressEvents (void)

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

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

  Helper method to set up and draw upgrade options.
- void __sendImprovementStateMessage (void)

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.15.1 Detailed Description

A settlement class (child class of TileImprovement).

4.15.2 Constructor & Destructor Documentation

4.15.2.1 WindTurbine()

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

Constructor for the WindTurbine class.

Ref: Wikipedia [2023]

Parameters

position_x	The x position of the tile.
position_y	The y position of the tile.
tile_resource	The renewable resource quality 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.

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

```
777     return;
778 }   /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

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

Destructor for the WindTurbine class.

```
1087 {
1088     std::cout « "WindTurbine at " « this « " destroyed" « std::endl;
1089
1090     return;
1091 }   /* ~WindTurbine() */
```

4.15.3 Member Function Documentation

4.15.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.15.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
274 {
275     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
276     std::default_random_engine generator(seed);
277
278     double mean =
279         this->tile_resource_scalar * MEAN_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
280
281     double stdev = STDEV_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
282
283     if (this->tile_resource_scalar > 1) {
284         stdev /= this->tile_resource_scalar;
285     }
286
287     std::normal_distribution<double> normal_dist(mean, stdev);
288
289     double capacity_factor = 0;
```

```
290
291     for (int i = 0; i < 30; i++) {
292         capacity_factor = normal_dist(generator);
293
294         if (capacity_factor < 0) {
295             capacity_factor = 0;
296         }
297
298         this->capacity_factor_vec[i] = capacity_factor;
299     }
300
301     return;
302 }   /* __computeCapacityFactors() */
```

4.15.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
345 {
346     double stored_energy_MWh = 0;
347     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
348
349     double demand_MWh = 0;
350     double production_MWh = 0;
351     double dispatch_MWh = 0;
352     double difference_MWh = 0;
353
354     double room_MWh = 0;
355
356     for (int i = 0; i < 30; i++) {
357         demand_MWh = this->demand_vec_MWh[i];
358         production_MWh = this->production_vec_MWh[i];
359
360         if (production_MWh <= demand_MWh) {
361             this->dispatch_vec_MWh[i] = production_MWh;
362             dispatch_MWh += this->dispatch_vec_MWh[i];
363
364             difference_MWh = demand_MWh - production_MWh;
365
366             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
367                 if (difference_MWh > stored_energy_MWh) {
368                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
369                     dispatch_MWh += stored_energy_MWh;
370                     stored_energy_MWh = 0;
371                 }
372
373                 else {
374                     this->dispatch_vec_MWh[i] += difference_MWh;
375                     dispatch_MWh += difference_MWh;
376                     stored_energy_MWh -= difference_MWh;
377                 }
378             }
379         }
380
381         else {
382             this->dispatch_vec_MWh[i] = demand_MWh;
383             dispatch_MWh += this->dispatch_vec_MWh[i];
384
385             difference_MWh = production_MWh - demand_MWh;
386
387             if (
388                 (storage_capacity_MWh > 0) and
389                 (stored_energy_MWh < storage_capacity_MWh)
390             ) {
391                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
392
393                 if (difference_MWh > room_MWh) {
394                     stored_energy_MWh += room_MWh;
395                 }
396
397                 else {
398                     stored_energy_MWh += difference_MWh;
399                 }
400             }
401         }
402     }
```

```
403
404     this->dispatchable_MWh = round(dispatch_MWh);
405
406     if (this->dispatch_MWh > this->dispatchable_MWh) {
407         this->dispatch_MWh = this->dispatch_MWh;
408     }
409
410     return;
411 }   /* __computeDispatch() */
```

4.15.3.4 __computeProduction()

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

Helper method to compute production values.

```
317 {
318     double production_MWh = 0;
319
320     for (int i = 0; i < 30; i++) {
321         this->production_vec_MWh[i] =
322             this->max_daily_production_MWh * this->capacity_factor_vec[i];
323
324         production_MWh += this->production_vec_MWh[i];
325     }
326
327     this->production_MWh = round(production_MWh);
328
329     return;
330 }   /* __computeProduction() */
```

4.15.3.5 __computeProductionCosts()

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

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

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

4.15.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
114 {
115     //  1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
```

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

4.15.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
552 {
553     //  1. draw power capacity upgrade sprite
554     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
555         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
556         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
557 
558         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
559         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
560 
561         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
562         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
563 
564         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
565 
566         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
567         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
568         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
569     }
570 
571     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
572 
573 
574     //  2. draw power capacity upgrade text
575     //                  16 char line = "                \n"
576     std::string power_upgrade_string = "POWER CAPACITY  \n";
577     power_upgrade_string            += "                \n";
578 
```

```
579     power_upgrade_string            += "CAPACITY:  ";
580     power_upgrade_string            += std::to_string(this->capacity_kW);
581     power_upgrade_string            += " kW\n";
582 
583     power_upgrade_string            += "LEVEL:     ";
584     power_upgrade_string            += std::to_string(this->upgrade_level);
585     power_upgrade_string            += "\n\n";
586 
587     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
588         power_upgrade_string        += "[W]: + 100 kW (";
589         power_upgrade_string        +=  std::to_string(WIND_TURBINE_BUILD_COST);
590         power_upgrade_string        += " K)\n";
591     }
592 
593     else {
594         power_upgrade_string        += " * MAX LEVEL *  \n";
595     }
596 
597     sf::Text power_upgrade_text = sf::Text(
598         power_upgrade_string,
599         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
600         16
601     );
602 
603     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
604     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
605     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
606 
607     this->render_window_ptr->draw(power_upgrade_text);
608 
609 
610     //  3. draw energy capacity (storage) upgrade sprite
611     this->render_window_ptr->draw(this->storage_upgrade_sprite);
612     this->render_window_ptr->draw(this->upgrade_plus_sprite);
613 
614 
615     //  4. draw energy capacity (storage) upgrade text
616     //                   16 char line = "               \n"
617     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
618     energy_upgrade_string            += "               \n";
619 
620     energy_upgrade_string            += "CAPACITY:  ";
621     energy_upgrade_string            += std::to_string(this->storage_level * 200);
622     energy_upgrade_string            += " kWh\n";
623 
624     energy_upgrade_string            += "LEVEL:     ";
625     energy_upgrade_string            += std::to_string(this->storage_level);
626     energy_upgrade_string            += "\n\n";
627 
628     if (this->storage_level < MAX_STORAGE_LEVELS) {
629         energy_upgrade_string        += "[D]: + 200 kWh (";
630         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
631         energy_upgrade_string        += " K)\n";
632     }
633 
634     else {
635         energy_upgrade_string += " * MAX LEVEL *  \n";
636     }
637 
638     sf::Text energy_upgrade_text = sf::Text(
639         energy_upgrade_string,
640         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
641         16
642     );
643 
644     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
645     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
646     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
647 
648     this->render_window_ptr->draw(energy_upgrade_text);
649 
650     return;
651 }   /* __drawUpgradeOptions() */
```

4.15.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
426 {
427     if (this->just_built) {
428         return;
429     }
430 
431     switch (this->event_ptr->key.code) {
432         case (sf::Keyboard::U): {
433             this->__openUpgradeMenu();
434 
435             break;
436         }
437 
438 
439         case (sf::Keyboard::W): {
440             if (this->production_menu_open) {
441                 this->dispatch_MWh++;
442 
443                 if (this->dispatch_MWh > this->dispatchable_MWh) {
444                     this->dispatch_MWh = 0;
445                 }
446 
447                 this->__computeProductionCosts();
448                 this->assets_manager_ptr->getSound("interface click")->play();
449             }
450 
451             else if (this->upgrade_menu_open) {
452                 this->__upgradePowerCapacity();
453             }
454 
455             break;
456         }
457 
458 
459         case (sf::Keyboard::S): {
460             if (this->production_menu_open) {
461                 this->dispatch_MWh--;
462 
463                 if (this->dispatch_MWh < 0) {
464                     this->dispatch_MWh = this->dispatchable_MWh;
465                 }
466 
467                 this->__computeProductionCosts();
468                 this->assets_manager_ptr->getSound("interface click")->play();
469             }
470 
471             break;
472         }
473 
474 
475         case (sf::Keyboard::D): {
476             if (this->upgrade_menu_open) {
477                 this->__upgradeStorageCapacity();
478                 this->__computeProduction();
479                 this->__computeDispatch();
480             }
481 
482             break;
483         }
484 
485 
486         default: {
487             // do nothing!
488 
489             break;
490         }
491     }
492 
493     return;
494 }   /* __handleKeyPressEvents() */
```

4.15.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
509 {
```

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

4.15.3.10 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

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

4.15.3.11 __setUpTileImprovementSpriteAnimated()

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

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

```
68 {
69     sf::Sprite diesel_generator_sheet(
70         *(this->assets_manager_ptr->getTexture("wind turbine"))
71     );
72
73     int n_elements = diesel_generator_sheet.getLocalBounds().height / 64;
74
75     for (int i = 0; i < n_elements; i++) {
76         this->tile_improvement_sprite_animated.push_back(
77             sf::Sprite(
78                 *(this->assets_manager_ptr->getTexture("wind turbine")),
```

```
79                 sf::IntRect(0, i * 64, 64, 64)
80             )
81         );
82 
83         this->tile_improvement_sprite_animated.back().setOrigin(
84             this->tile_improvement_sprite_animated.back().getLocalBounds().width / 2,
85             this->tile_improvement_sprite_animated.back().getLocalBounds().height
86         );
87 
88         this->tile_improvement_sprite_animated.back().setPosition(
89             this->position_x,
90             this->position_y - 32
91         );
92 
93         this->tile_improvement_sprite_animated.back().setColor(
94             sf::Color(255, 255, 255, 0)
95         );
96     }
97 
98     return;
99 }   /* __setUpTileImprovementSpriteAnimated() */
```

4.15.3.12 __upgradePowerCapacity()

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

Helper method to upgrade the power capacity.

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

4.15.3.13 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

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

4.15.3.14 draw()

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

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

Reimplemented from TileImprovement.

```
985 {
986     //  1. if just built, call base method and return
987     if (this->just_built) {
988         TileImprovement :: draw();
989 
990         return;
991     }
992 
993 
994     //  2. handle upgrade effects
995     if (this->just_upgraded) {
996         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
997             this->tile_improvement_sprite_animated[i].setColor(
998                 sf::Color(
999                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1000                     255,
1001                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1002                     255
1003                 )
1004             );
1005 
1006             this->tile_improvement_sprite_animated[i].setScale(
1007                 sf::Vector2f(
1008                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1009                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1010                 )
1011             );
1012         }
1013 
1014         this->upgrade_frame++;
1015     }
1016 
1017     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1018         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1019             this->tile_improvement_sprite_animated[i].setColor(
1020                 sf::Color(255,255,255,255)
1021             );
1022 
1023             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1024         }
```

```
1025
1026         this->just_upgraded = false;
1027         this->upgrade_frame = 0;
1028     }
1029
1030
1031     //  3. draw first element of animated sprite
1032     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1033
1034
1035     //  4. draw second element of animated sprite
1036     if (this->is_running) {
1037         //...
1038     }
1039
1040     else {
1041         //...
1042     }
1043
1044     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1045
1046
1047     //  5. draw storage upgrades
1048     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1049         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1050     }
1051
1052
1053     //  6. draw production menu
1054     if (this->production_menu_open) {
1055         this->render_window_ptr->draw(this->production_menu_backing);
1056         this->render_window_ptr->draw(this->production_menu_backing_text);
1057
1058         this->__drawProductionMenu();
1059     }
1060
1061
1062     //  7. draw upgrade menu
1063     if (this->upgrade_menu_open) {
1064         this->render_window_ptr->draw(this->upgrade_menu_backing);
1065         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1066
1067         this->__drawUpgradeOptions();
1068     }
1069
1070     this->frame++;
1071     return;
1072 }   /* draw() */
```

4.15.3.15 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
795 {
796     //                  32 char x 17 line console "--------------------------------\n";
797     std::string options_substring                = "CAPACITY:      ";
798     options_substring                           += std::to_string(this->capacity_kW);
799     options_substring                           += " kW (level ";
800     options_substring                           += std::to_string(this->upgrade_level);
801     options_substring                           += ")\n";
802
803     options_substring                           += "PRODUCTION:    ";
804     options_substring                           += std::to_string(this->production_MWh);
805     options_substring                           += " MWh\n";
806
807     options_substring                           += "DISPATCHABLE:  ";
```

```
808     options_substring                           += std::to_string(this->dispatchable_MWh);
809     options_substring                           += " MWh\n";
810
811     options_substring                           += "HEALTH:      ";
812     options_substring                           += std::to_string(this->health);
813     options_substring                           += "/100";
814
815     if (this->health <= 0) {
816         options_substring                       += " ** BROKEN! **\n";
817     }
818
819     else {
820         options_substring                       += "\n";
821     }
822
823     options_substring                           += "                                \n";
824     options_substring                           += " **** WIND TURBINE OPTIONS **** \n";
825     options_substring                           += "                                \n";
826     options_substring                           += "     [E]:  OPEN PRODUCTION MENU \n";
827     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
828     options_substring                           += "HOLD [P]:  SCRAP (";
829     options_substring                           += std::to_string(SCRAP_COST);
830     options_substring                           += " K)";
831
832     return options_substring;
833 }   /* getTileOptionsSubstring() */
```

4.15.3.16 processEvent()

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

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

Reimplemented from TileImprovement.

```
936 {
937     TileImprovement :: processEvent();
938
939     if (this->event_ptr->type == sf::Event::KeyPressed) {
940         this->__handleKeyPressEvents();
941     }
942
943     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
944         this->__handleMouseButtonEvents();
945     }
946
947     return;
948 }   /* processEvent() */
```

4.15.3.17 processMessage()

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

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

Reimplemented from TileImprovement.

```
963 {
964     TileImprovement :: processMessage();
965
966     //...
967
968     return;
969 }   /* processMessage() */
```

4.15.3.18 setIsSelected()

```
void WindTurbine::setIsSelected (
            bool is_selected )  [virtual]
```

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

```
850 {
851     TileImprovement :: setIsSelected(is_selected);
852 
853     if (this->is_running and this->is_selected) {
854         this->assets_manager_ptr->getSound("wind turbine running")->play();
855     }
856 
857     return;
858 }   /* setIsSelected() */
```

4.15.3.19 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
914 {
915     this->__computeCapacityFactors();
916     this->__computeProduction();
917     this->__computeProductionCosts();
918     this->__computeDispatch();
919 
920     return;
921 }   /* update() */
```

4.15.4 Member Data Documentation

4.15.4.1 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.15.4.2 capacity_kW

```
int WindTurbine::capacity_kW
```

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

4.15.4.3 dispatch_MWh

```
int WindTurbine::dispatch_MWh
```

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

4.15.4.4 dispatch_vec_MWh

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

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

4.15.4.5 dispatchable_MWh

```
int WindTurbine::dispatchable_MWh
```

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

4.15.4.6 max_daily_production_MWh

```
double WindTurbine::max_daily_production_MWh
```

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

4.15.4.7 production_MWh

```
int WindTurbine::production_MWh
```

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

4.15.4.8 production_vec_MWh

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

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

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

- header/WindTurbine.h
- source/WindTurbine.cpp

Chapter 5

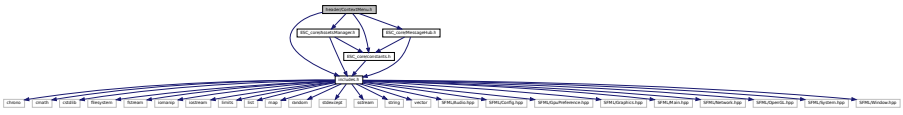
File Documentation

5.1 header/ContextMenu.h File Reference

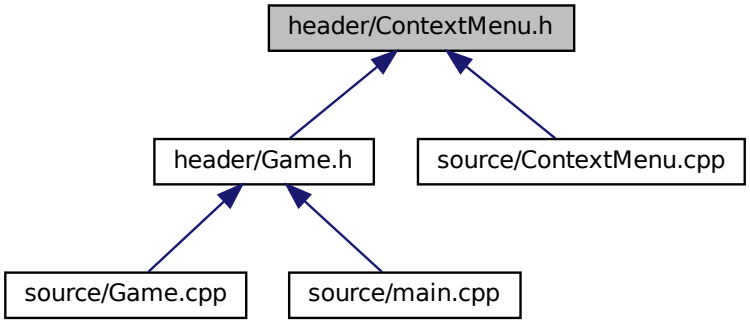
Header file for the ContextMenu class.

```
#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 };
```

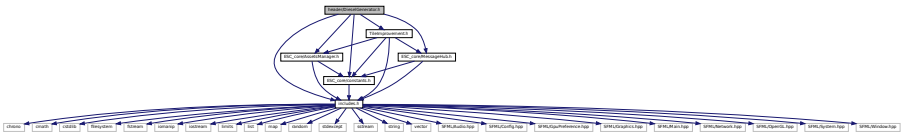
5.2 header/DieselGenerator.h File Reference

Header file for the DieselGenerator class.

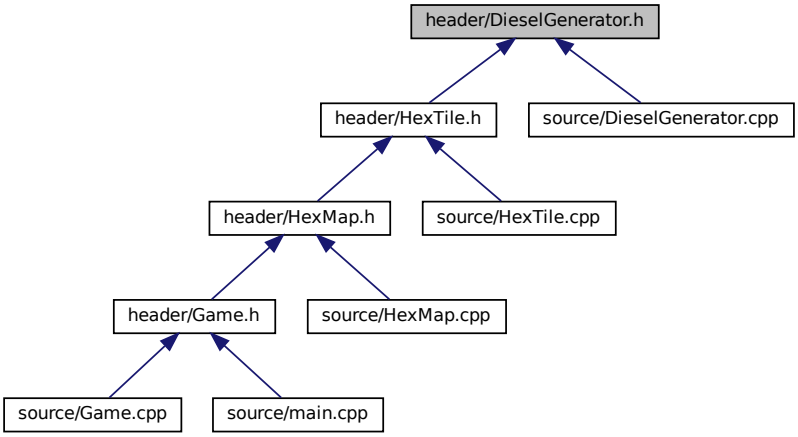
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
```

```
#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

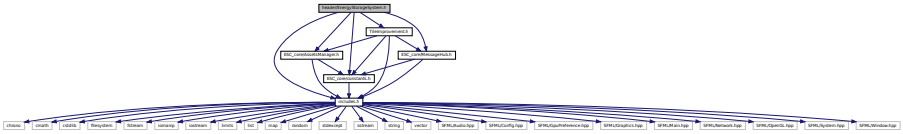
Header file for the DieselGenerator class.

5.3 header/EnergyStorageSystem.h File Reference

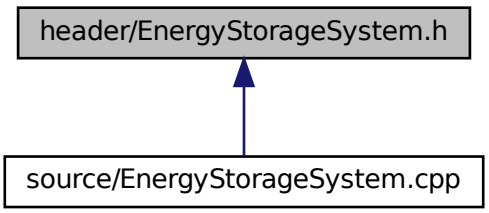
Header file for the EnergyStorageSystem class.

```
#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).

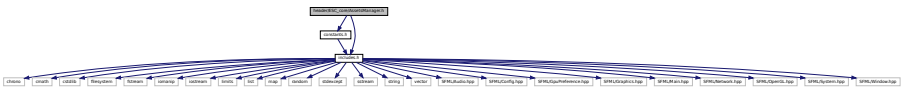
5.3.1 Detailed Description

Header file for the EnergyStorageSystem class.

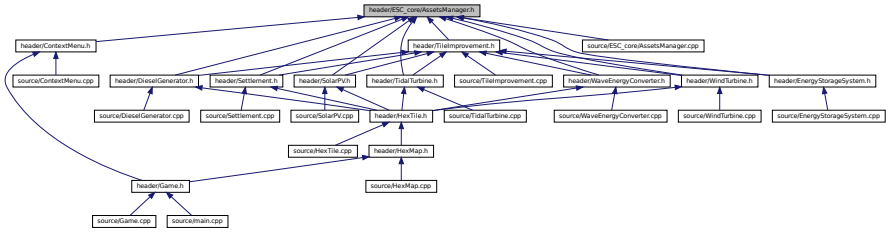
5.4 header/ESC_core/AssetsManager.h File Reference

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.

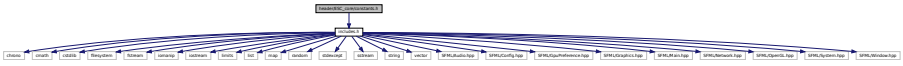
5.4.1 Detailed Description

Header file for the AssetsManager class.

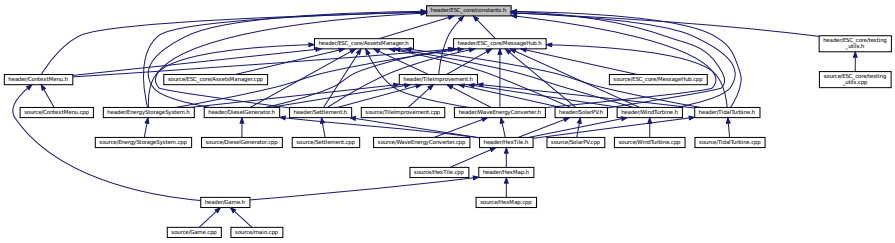
5.5 header/ESC_core/constants.h File Reference

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 std::string SETTLEMENT_CHANNEL = "SETTLEMENT CHANNEL"

  A message channel for the settlement.
- const int CLEAR_FOREST_COST = 40

  The cost of clearing a forest tile.
- const int CLEAR_MOUNTAINS_COST = 250

  The cost of clearing a mountains tile.
- const int CLEAR_PLAINS_COST = 20

  The cost of clearing a plains tile.
- const int DIESEL_GENERATOR_BUILD_COST = 100

  The cost of building (or ugrading) a diesel generator in 100 kW increments.
- const int WIND_TURBINE_BUILD_COST = 400

  The cost of building (or upgrading) a wind turbine in 100 kW increments.
- const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25

  The additional cost of building on water.
- const int SOLAR_PV_BUILD_COST = 300

  The cost of building (or upgrading) a solar PV array in 100 kW increments.
- const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5

  The additional cost of building on water.
- const int TIDAL_TURBINE_BUILD_COST = 600

  The cost of building (or upgrading) a tidal turbine in 100 kW increments.
- const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800

  The cost of building (or upgrading) a wave energy converter in 100 kW increments.
- const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 160

  The cost of adding energy storage in 200 kWh increments.
- const int SCRAP_COST = 50

  The cost of scrapping a tile improvement (other than settlement).
- const int MAX_UPGRADE_LEVELS = 5

  The maximum upgrade level of any tile improvement.
- const int MAX_STORAGE_LEVELS = 5

  The maximum storage level of any tile improvement.
- const int STARTING_CREDITS = 999999
- const double CREDITS_PER_MWH_SERVED = 1

  The number of credits (x1000) earned.
- 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 POPULATION_MONTHLY_GROWTH_RATE = 1.005

  The monthly population growth rate.
- const double LITRES_DIESEL_PER_MWH_PRODUCTION = 373.175

  The litres of diesel consumed in producing 1 MWh (assumes higher heating value and constant thermal efficiency of 0.25).
- const double COST_PER_LITRE_DIESEL = 1.70

  The cost of a litre of diesel.

- const double KG_CO2E_PER_LITRE_DIESEL = 3.1596

  The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.
- const double DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenace cost of running a diesel generator (assumed 0.05 credits per kWh produced).
- const double SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION = 10

  The operation and maintenance cost of running a solar PV array (assumed 0.01 credits per kWh produced).
- const double TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a tidal turbine (assumed 0.05 credits per kWh produced).
- const double WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a wave energy converter (assumed 0.05 credits per kWh produced).
- const double WIND_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a wind turbine (assumed 0.05 credits per kWh produced).
- const std::vector< double > MEAN_DAILY_DEMAND_RATIOS

  The mean daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_DEMAND_RATIOS

  The standard deviation in daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const double MAXIMUM_DAILY_DEMAND_PER_CAPITA = 0.0475

  The maximum daily demand [MWh] (at any point in the year) per capita.
- const std::vector< double > MEAN_DAILY_SOLAR_CAPACITY_FACTORS

  The mean daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_SOLAR_CAPACITY_FACTORS

  The standard deviation in daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const double DAILY_TIDAL_CAPACITY_FACTOR = 0.2175

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

  The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_WAVE_CAPACITY_FACTORS

  The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > MEAN_DAILY_WIND_CAPACITY_FACTORS

  The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_WIND_CAPACITY_FACTORS

  The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::string GAME_CHANNEL = "GAME CHANNEL"

  A message channel for game messages.
- const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"

  A message channel for game state messages.

5.5.1 Detailed Description

Header file for various constants.

5.5.2 Function Documentation

5.5.2.1 FOREST_GREEN()

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

The base colour of a forest tile.

5.5.2.2 LAKE_BLUE()

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

The base colour of a lake (water) tile.

5.5.2.3 MENU_FRAME_GREY()

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

The base colour of the context menu frame.

5.5.2.4 MONOCHROME_SCREEN_BACKGROUND()

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

The base colour of old monochrome screens.

5.5.2.5 MONOCHROME_TEXT_AMBER()

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

The base colour of old monochrome text (amber).

5.5.2.6 MONOCHROME_TEXT_GREEN()

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

The base colour of old monochrome text (green).

5.5.2.7 MONOCHROME_TEXT_RED()

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

The base colour of old monochrome text (red).

5.5.2.8 MOUNTAINS_GREY()

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

The base colour of a mountains tile.

5.5.2.9 OCEAN_BLUE()

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

The base colour of an ocean (water) tile.

5.5.2.10 PLAINS_YELLOW()

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

The base colour of a plains tile.

5.5.2.11 RESOURCE_CHIP_GREY()

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

The base colour of the resource chip (backing).

5.5.2.12 VISUAL_SCREEN_FRAME_GREY()

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

The base colour of the framing of the visual screen.

5.5.3 Variable Documentation

5.5.3.1 BUILD_SETTLEMENT_COST

```
const int BUILD_SETTLEMENT_COST = 250
```

The cost of building a settlement.

5.5.3.2 CLEAR_FOREST_COST

```
const int CLEAR_FOREST_COST = 40
```

The cost of clearing a forest tile.

5.5.3.3 CLEAR_MOUNTAINS_COST

```
const int CLEAR_MOUNTAINS_COST = 250
```

The cost of clearing a mountains tile.

5.5.3.4 CLEAR_PLAINS_COST

```
const int CLEAR_PLAINS_COST = 20
```

The cost of clearing a plains tile.

5.5.3.5 COST_PER_LITRE_DIESEL

```
const double COST_PER_LITRE_DIESEL = 1.70
```

The cost of a litre of diesel.

5.5.3.6 CREDITS_PER_MWH_SERVED

```
const double CREDITS_PER_MWH_SERVED = 1
```

The number of credits (x1000) earned.

5.5.3.7 DAILY_TIDAL_CAPACITY_FACTOR

```
const double DAILY_TIDAL_CAPACITY_FACTOR = 0.2175
```

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

5.5.3.8 DIESEL_GENERATOR_BUILD_COST

```
const int DIESEL_GENERATOR_BUILD_COST = 100
```

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

5.5.3.9 DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

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

5.5.3.10 EMISSIONS_LIFETIME_LIMIT_TONNES

```
const int EMISSIONS_LIFETIME_LIMIT_TONNES = 1500
```

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

5.5.3.11 ENERGY_STORAGE_SYSTEM_BUILD_COST

```
const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 160
```

The cost of adding energy storage in 200 kWh increments.

5.5.3.12 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

5.5.3.13 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

5.5.3.14 GAME_CHANNEL

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

A message channel for game messages.

5.5.3.15 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

5.5.3.16 GAME_STATE_CHANNEL

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

A message channel for game state messages.

5.5.3.17 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.18 HEX_MAP_CHANNEL

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

A message channel for hex map messages.

5.5.3.19 KG_CO2E_PER_LITRE_DIESEL

```
const double KG_CO2E_PER_LITRE_DIESEL = 3.1596
```

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

5.5.3.20 LITRES_DIESEL_PER_MWH_PRODUCTION

```
const double LITRES_DIESEL_PER_MWH_PRODUCTION = 373.175
```

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

5.5.3.21 MAX_STORAGE_LEVELS

```
const int MAX_STORAGE_LEVELS = 5
```

The maximum storage level of any tile improvement.

5.5.3.22 MAX_UPGRADE_LEVELS

```
const int MAX_UPGRADE_LEVELS = 5
```

The maximum upgrade level of any tile improvement.

5.5.3.23 MAXIMUM_DAILY_DEMAND_PER_CAPITA

```
const double MAXIMUM_DAILY_DEMAND_PER_CAPITA = 0.0475
```

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

5.5.3.24 MEAN_DAILY_DEMAND_RATIOS

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

Initial value:

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

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

5.5.3.25 MEAN_DAILY_SOLAR_CAPACITY_FACTORS

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

Initial value:

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

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

5.5.3.26 MEAN_DAILY_WAVE_CAPACITY_FACTORS

```
const std::vector<double> MEAN_DAILY_WAVE_CAPACITY_FACTORS
```

Initial value:

```
= {
    0.742, 0.694, 0.618,
    0.467, 0.366, 0.292,
    0.280, 0.293, 0.374,
    0.424, 0.662, 0.600
}
```

The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.27 MEAN_DAILY_WIND_CAPACITY_FACTORS

```
const std::vector<double> MEAN_DAILY_WIND_CAPACITY_FACTORS
```

Initial value:

```
= {
    0.591, 0.594, 0.627,
    0.629, 0.579, 0.537,
    0.442, 0.507, 0.587,
    0.618, 0.611, 0.580
}
```

The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.28 NO_TILE_SELECTED_CHANNEL

```
const std::string NO_TILE_SELECTED_CHANNEL = "NO TILE SELECTED CHANNEL"
```

A message channel for no tile selected messages.

5.5.3.29 POPULATION_MONTHLY_GROWTH_RATE

```
const double POPULATION_MONTHLY_GROWTH_RATE = 1.005
```

The monthly population growth rate.

5.5.3.30 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.31 SCRAP_COST

```
const int SCRAP_COST = 50
```

The cost of scrapping a tile improvement (other than settlement).

5.5.3.32 SECONDS_PER_FRAME

```
const double SECONDS_PER_FRAME = 1.0 / 60
```

Target seconds per frame (just reciprocal of target frames per second).

5.5.3.33 SECONDS_PER_MONTH

```
const unsigned long long int SECONDS_PER_MONTH = 2628164
```

5.5.3.34 SECONDS_PER_YEAR

```
const unsigned long long int SECONDS_PER_YEAR = 31537970
```

5.5.3.35 SETTLEMENT_CHANNEL

```
const std::string SETTLEMENT_CHANNEL = "SETTLEMENT CHANNEL"
```

A message channel for the settlement.

5.5.3.36 SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION = 10
```

The operation and maintenance cost of running a solar PV array (assumed 0.01 credits per kWh produced).

5.5.3.37 SOLAR_PV_BUILD_COST

```
const int SOLAR_PV_BUILD_COST = 300
```

The cost of building (or upgrading) a solar PV array in 100 kW increments.

5.5.3.38 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.5
```

The additional cost of building on water.

5.5.3.39 STARTING_CREDITS

```
const int STARTING_CREDITS = 999999
```

5.5.3.40 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.41 STDEV_DAILY_DEMAND_RATIOS

```
const std::vector<double> STDEV_DAILY_DEMAND_RATIOS
```

Initial value:

```
= {
    0.069, 0.074, 0.072,
    0.072, 0.063, 0.060,
    0.012, 0.031, 0.040,
    0.049, 0.063, 0.053
}
```

The standard deviation in daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.

5.5.3.42 STDEV_DAILY_SOLAR_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_SOLAR_CAPACITY_FACTORS
```

Initial value:

```
= {
    0.013, 0.024, 0.043,
    0.049, 0.072, 0.072,
    0.076, 0.065, 0.048,
    0.026, 0.018, 0.009
}
```

The standard deviation in daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.43 STDEV_DAILY_WAVE_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_WAVE_CAPACITY_FACTORS
```

Initial value:

```
= {
    0.146, 0.135, 0.163,
    0.145, 0.158, 0.106,
    0.086, 0.058, 0.145,
    0.171, 0.184, 0.309
}
```

The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.44 STDEV_DAILY_WIND_CAPACITY_FACTORS

```
const std::vector<double> STDEV_DAILY_WIND_CAPACITY_FACTORS
```

Initial value:

```
= {
    0.147, 0.142, 0.198,
    0.154, 0.162, 0.202,
    0.180, 0.217, 0.198,
    0.168, 0.141, 0.168
}
```

The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.45 TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a tidal turbine (assumed 0.05 credits per kWh produced).

5.5.3.46 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 600
```

The cost of building (or upgrading) a tidal turbine in 100 kW increments.

5.5.3.47 TILE_RESOURCE_CUMULATIVE_PROBABILITIES

```
const std::vector<double> TILE_RESOURCE_CUMULATIVE_PROBABILITIES
```

Initial value:

```
= {
    0.10,
    0.30,
    0.70,
    0.90,
    1.00
}
```

Cumulative probabilities for each tile resource (to support procedural generation).

5.5.3.48 TILE_SELECTED_CHANNEL

```
const std::string TILE_SELECTED_CHANNEL = "TILE SELECTED CHANNEL"
```

A message channel for tile selection messages.

5.5.3.49 TILE_STATE_CHANNEL

```
const std::string TILE_STATE_CHANNEL = "TILE STATE CHANNEL"
```

A message channel for tile state messages.

5.5.3.50 TILE_TYPE_CUMULATIVE_PROBABILITIES

```
const std::vector<double> TILE_TYPE_CUMULATIVE_PROBABILITIES
```

Initial value:

```
= {
    0.25,
    0.50,
    0.75,
    1.00
}
```

Cumulative probabilities for each tile type (to support procedural generation).

5.5.3.51 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 800
```

The cost of building (or upgrading) a wave energy converter in 100 kW increments.

5.5.3.52 WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a wave energy converter (assumed 0.05 credits per kWh produced).

5.5.3.53 WIND_OP_MAINT_COST_PER_MWH_PRODUCTION

```
const double WIND_OP_MAINT_COST_PER_MWH_PRODUCTION = 50
```

The operation and maintenance cost of running a wind turbine (assumed 0.05 credits per kWh produced).

5.5.3.54 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 400
```

The cost of building (or upgrading) a wind turbine in 100 kW increments.

5.5.3.55 WIND_TURBINE_WATER_BUILD_MULTIPLIER

```
const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.25
```

The additional cost of building on water.

5.6 header/ESC_core/doxygen_cite.h File Reference

Header file which simply cites the doxygen tool.

5.6.1 Detailed Description

Header file which simply cites the doxygen tool.

Ref: van Heesch. [2023]

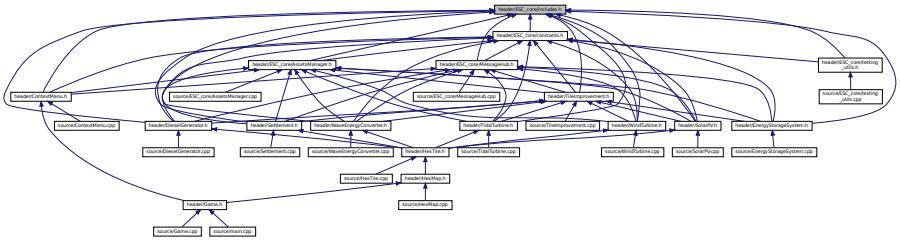
5.7 header/ESC_core/includes.h File Reference

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 <random>
#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.

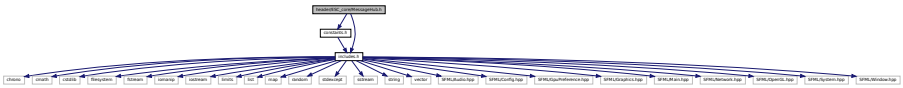
Ref: Gomila [2023]

5.8 header/ESC_core/MessageHub.h File Reference

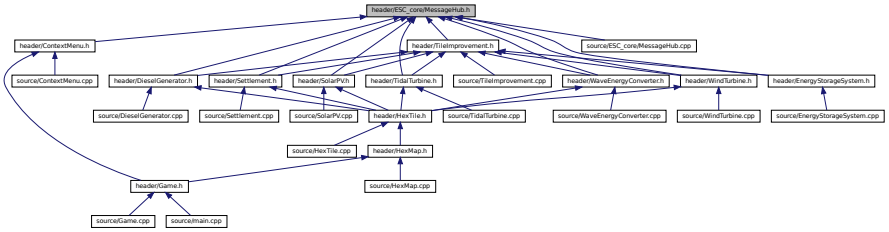
Header file for the MessageHub class.

```
#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

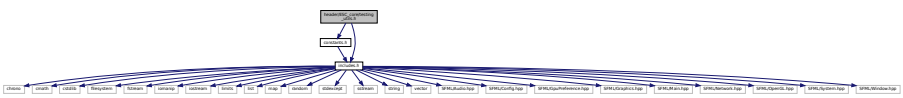
Header file for the MessageHub class.

5.9 header/ESC_core/testing_utils.h File Reference

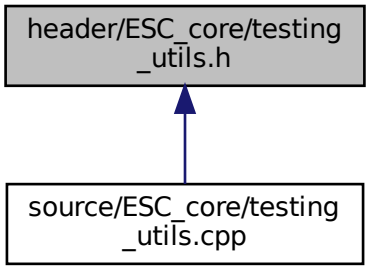
Header file for various testing utilities.

```
#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.

Parameters

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

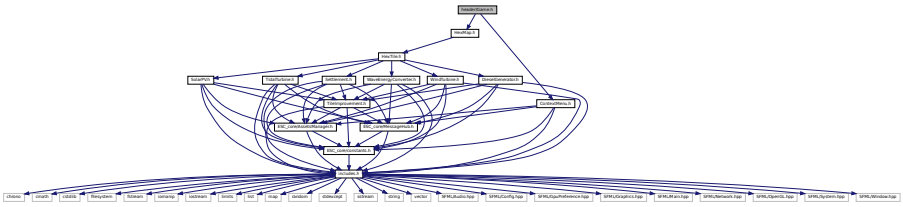
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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() */
```

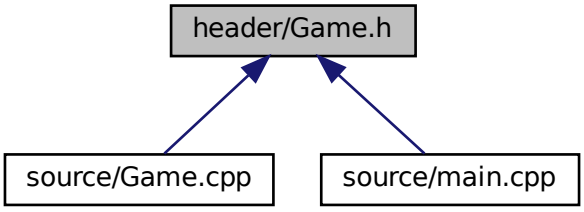
5.10 header/Game.h File Reference

```
#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.

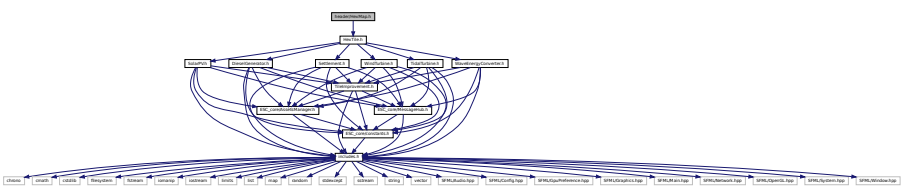
```
66                {
67      BUILD_SETTLEMENT,
68      SYSTEM_MANAGEMENT,
69      LOSS_EMISSIONS,
70      LOSS_DEMAND,
71      LOSS_CREDITS,
72      VICTORY,
73      N_GAME_PHASES
74  };  /* GamePhase */
```

5.11 header/HexMap.h File Reference

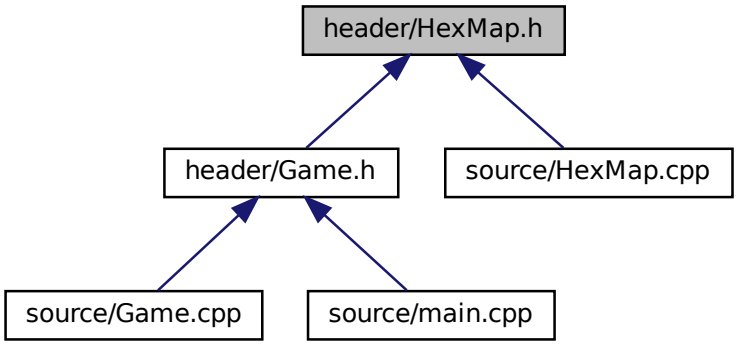
Header file for the HexMap class.

```
#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

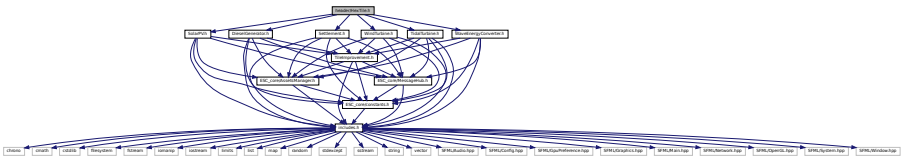
Header file for the HexMap class.

5.12 header/HexTile.h File Reference

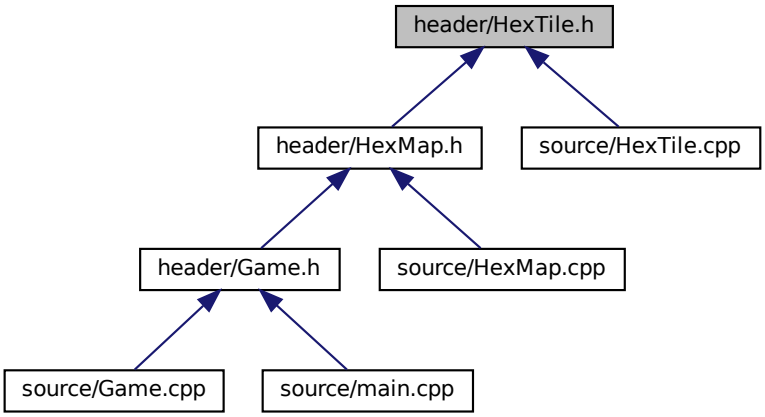
Header file for the Game class.

```
#include "DieselGenerator.h"
#include "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 */
```

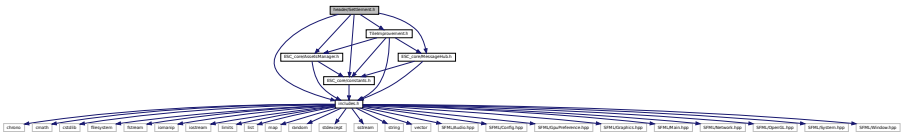
5.13 header/Settlement.h File Reference

Header file for the Settlement class.

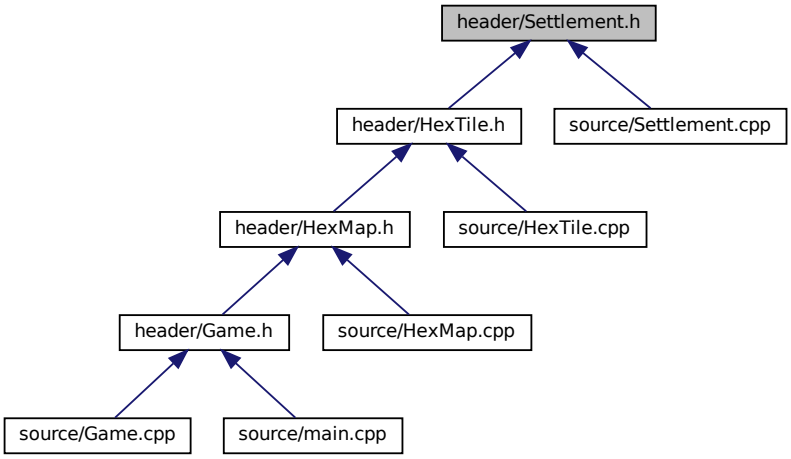
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

```
#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.

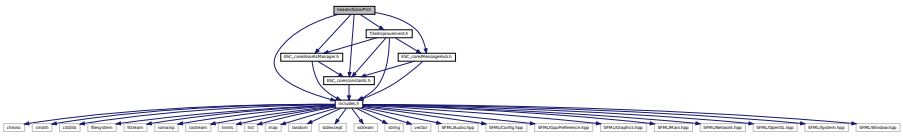
5.14 header/SolarPV.h File Reference

Header file for the SolarPV class.

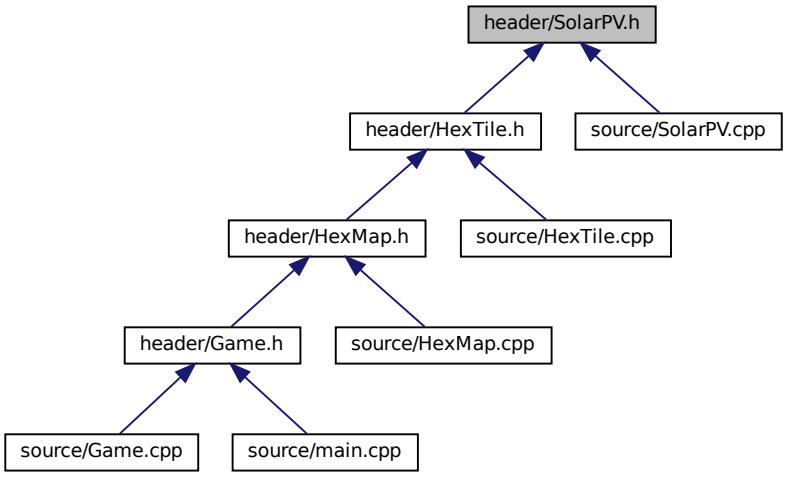
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

```
#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.

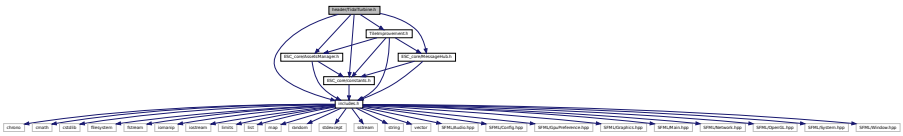
5.15 header/TidalTurbine.h File Reference

Header file for the TidalTurbine class.

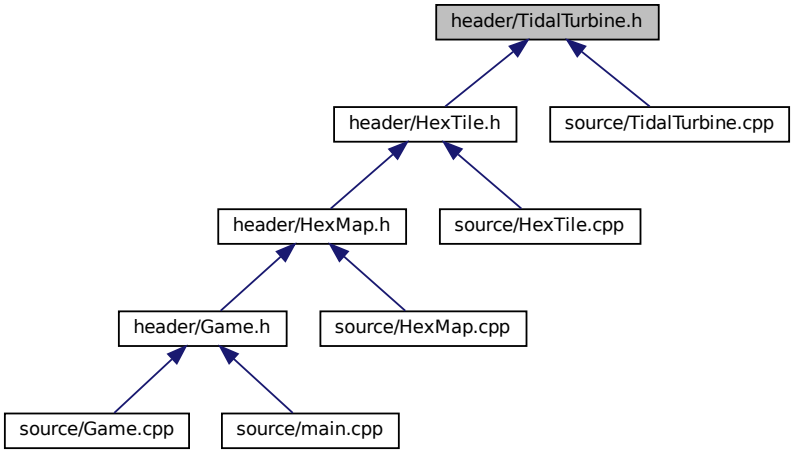
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
#include "ESC_core/MessageHub.h"
```

```
#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.

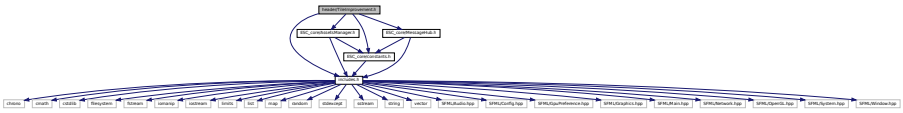
5.16 header/TileImprovement.h File Reference

Header file for the TileImprovement class.

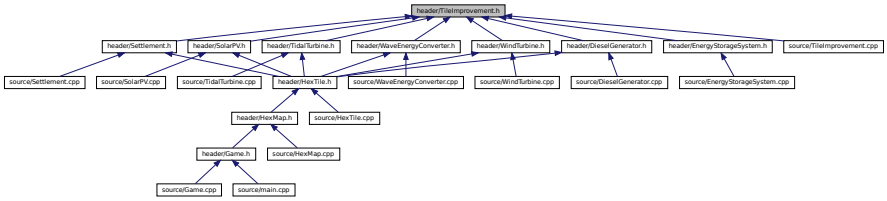
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
```

```
#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.

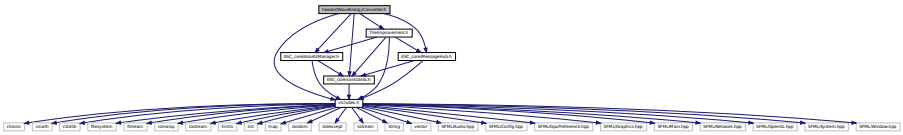
```
68                          {
69     SETTLEMENT,
70     DIESEL_GENERATOR,
71     SOLAR_PV,
72     WIND_TURBINE,
73     TIDAL_TURBINE,
74     WAVE_ENERGY_CONVERTER,
75     ENERGY_STORAGE_SYSTEM,
76     N_TILE_IMPROVEMENT_TYPES
77 };  /* TileImprovementType */
```

5.17 header/WaveEnergyConverter.h File Reference

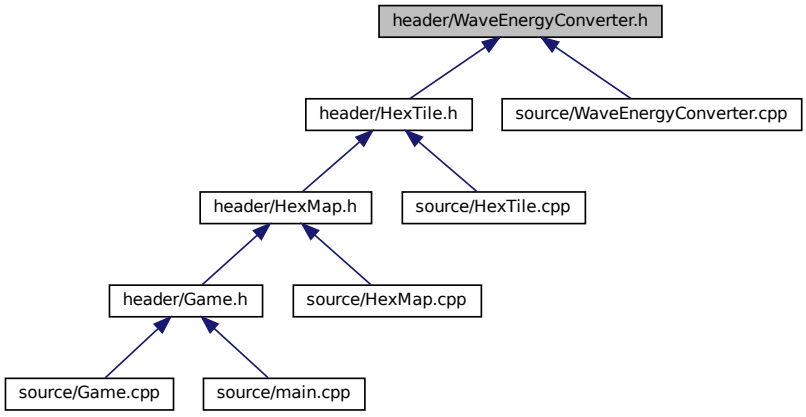
Header file for the WaveEnergyConverter class.

```
#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

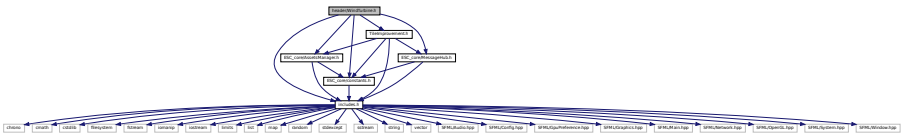
Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

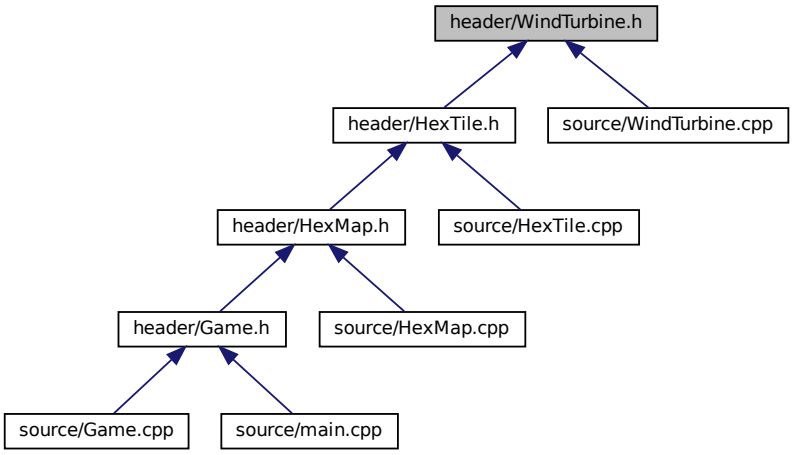
Header file for the WindTurbine class.

```
#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).

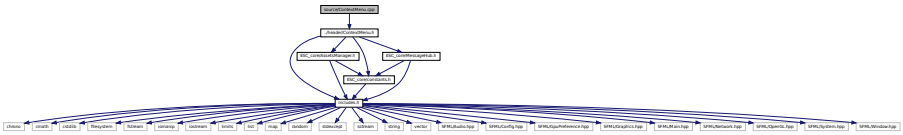
5.18.1 Detailed Description

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

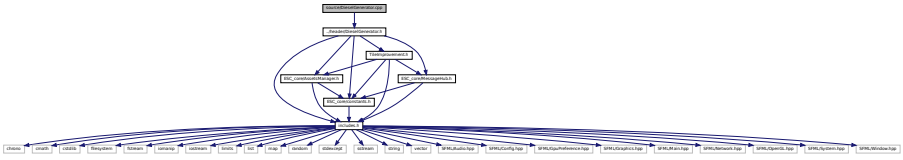
Implementation file for the ContextMenu class.

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

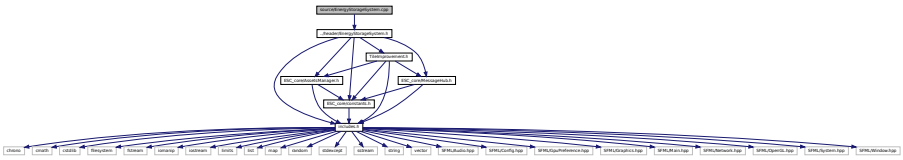
Implementation file for the DieselGenerator class.

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

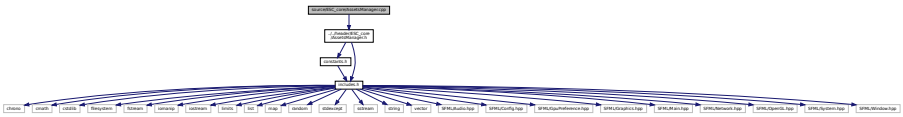
Implementation file for the EnergyStorageSystem class.

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

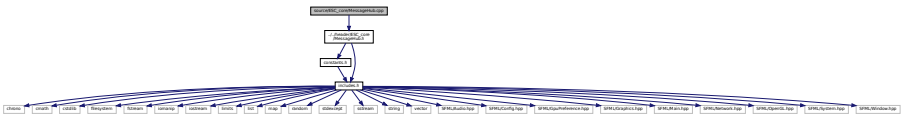
Implementation file for the AssetsManager class.

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.

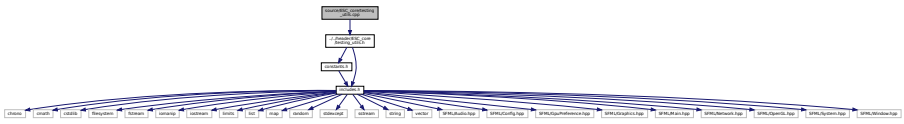
A class which acts as a central hub for inter-object message traffic.

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

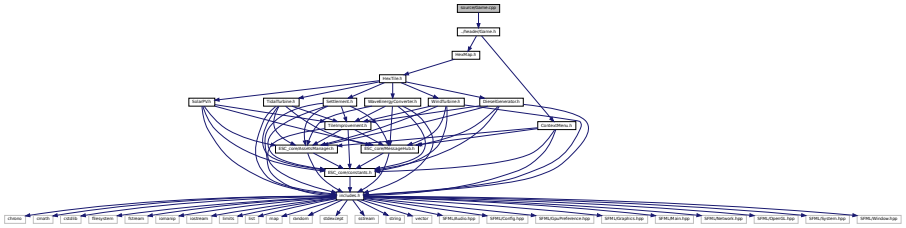
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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

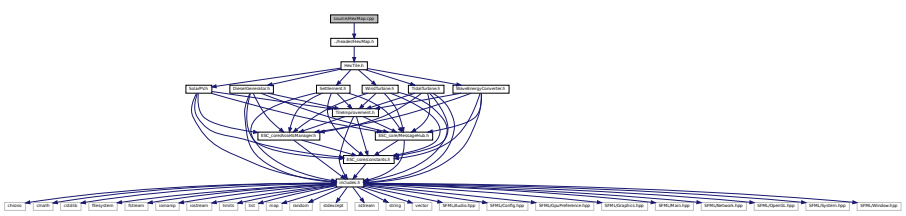
Implementation file for the Game class.

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

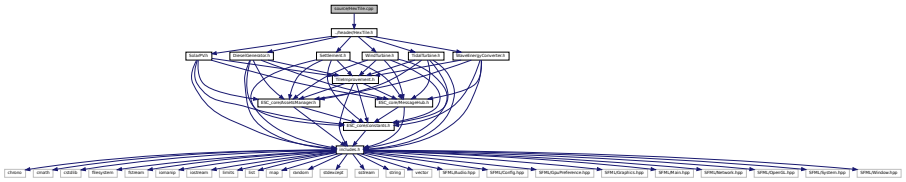
Implementation file for the HexMap class.

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

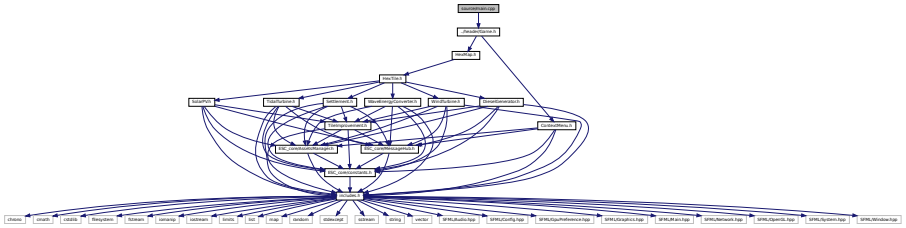
Implementation file for the HexTile class.

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.

```
329 {
330     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
331         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
332         "Road To Zero"
333     );
334 
335     return render_window_ptr;
336 }   /* constructRenderWindow() */
```

5.28.2.2 loadAssets()

```
void loadAssets (
            AssetsManager * assets_manager_ptr )
```

Helper function to load game assets.

Parameters

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     assets_manager_ptr->loadTexture(
149         "assets/tile_sheets/upgrade_arrow_16x16_1_CC-BY.png",
150         "upgrade arrow"
151     );
152
153     assets_manager_ptr->loadTexture(
154         "assets/tile_sheets/upgrade_plus_16x16_1_CC-BY.png",
155         "upgrade plus"
156     );
157
158     assets_manager_ptr->loadTexture(
159         "assets/tile_sheets/energy_storage_16x16_1_CC-BY.png",
160         "storage level"
161     );
162
163     assets_manager_ptr->loadTexture(
164         "assets/tile_sheets/coin_16x16_1_CC-BY.png",
165         "coin"
166     );
167
168
169     //  3. load sounds
170     assets_manager_ptr->loadSound(
171         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
```

```
172         "coin ring"
173     );
174
175     assets_manager_ptr->loadSound(
176         "assets/audio/samples/mixkit-positive-notification-951_MixkitFree.ogg",
177         "positive notification"
178     );
179
180     assets_manager_ptr->loadSound(
181         "assets/audio/samples/mixkit-sci-fi-click-900_MixkitFree.ogg",
182         "sci-fi click"
183     );
184
185     assets_manager_ptr->loadSound(
186         "assets/audio/samples/mixkit-apartment-buzzer-bell-press-932_MixkitFree.ogg",
187         "insufficient credits"
188     );
189
190     assets_manager_ptr->loadSound(
191         "assets/audio/samples/mixkit-data-scanner-2487_MixkitFree.ogg",
192         "resource assessment"
193     );
194
195     assets_manager_ptr->loadSound(
196         "assets/audio/samples/mixkit-interface-click-1126_MixkitFree.ogg",
197         "console string print"
198     );
199
200     assets_manager_ptr->loadSound(
201         "assets/audio/samples/mixkit-video-game-retro-click-237_MixkitFree.ogg",
202         "resource overlay toggle on"
203     );
204
205     assets_manager_ptr->loadSound(
206         "assets/audio/samples/mixkit-video-game-retro-click-237_REVERSED_MixkitFree.ogg",
207         "resource overlay toggle off"
208     );
209
210     assets_manager_ptr->loadSound(
211         "assets/audio/samples/mixkit-explosion-with-rocks-debris-1703_MixkitFree.ogg",
212         "clear mountains tile"
213     );
214
215     assets_manager_ptr->loadSound(
216         "assets/audio/samples/mixkit-arcade-game-explosion-2759_MixkitFree.ogg",
217         "clear non-mountains tile"
218     );
219
220     assets_manager_ptr->loadSound(
221         "assets/audio/samples/mixkit-electronic-retro-block-hit-2185_MixkitFree.ogg",
222         "place improvement"
223     );
224
225     assets_manager_ptr->loadSound(
226         "assets/audio/samples/mixkit-video-game-lock-2851_REVERSED_MixkitFree.ogg",
227         "build menu open"
228     );
229
230     assets_manager_ptr->loadSound(
231         "assets/audio/samples/mixkit-video-game-lock-2851_MixkitFree.ogg",
232         "build menu close"
233     );
234
235     assets_manager_ptr->loadSound(
236         "assets/audio/samples/mixkit-jump-into-the-water-1180_MixkitFree.ogg",
237         "splash"
238     );
239
240     assets_manager_ptr->loadSound(
241         "assets/audio/samples/505316__nuncaconoci__diesel_CC0.ogg",
242         "diesel running"
243     );
244
245     assets_manager_ptr->loadSound(
246         "assets/audio/samples/33460__pempi__320d_2_CC-BY.ogg",
247         "diesel start"
248     );
249
250     assets_manager_ptr->loadSound(
251         "assets/audio/samples/132724__andy_gardner__wind-turbine-blades_CC-BY.ogg",
252         "wind turbine running"
253     );
254
255     assets_manager_ptr->loadSound(
256         "assets/audio/samples/58416__darren1979__oceanwaves_CC-SAMPLING.ogg",
257         "ocean waves"
258     );
```

```
259
260     assets_manager_ptr->loadSound(
261         "assets/audio/samples/369927__mephisto_egmont__water-flowing-in-tubes_CC-BY.ogg",
262         "water flow"
263     );
264
265     assets_manager_ptr->loadSound(
266
      "assets/audio/samples/647663__jotraing__electric-train-motor-idle-loop-new-generation-rollingstock_CC0.ogg",
267         "solar hum"
268     );
269
270     assets_manager_ptr->loadSound(
271         "assets/audio/samples/mixkit-epic-futuristic-movie-accent-2913_MixkitFree.ogg",
272         "game title screen"
273     );
274
275     assets_manager_ptr->loadSound(
276         "assets/audio/samples/mixkit-calm-park-with-people-and-children_MixkitFree.ogg",
277         "people and children"
278     );
279
280     assets_manager_ptr->loadSound(
281         "assets/audio/samples/mixkit-magical-coin-win-1936_MixkitFree.ogg",
282         "upgrade"
283     );
284
285     assets_manager_ptr->loadSound(
286         "assets/audio/samples/mixkit-cool-interface-click-tone-2568_MixkitFree.ogg",
287         "interface click"
288     );
289
290     assets_manager_ptr->loadSound(
291         "assets/audio/samples/mixkit-factory-metal-hard-hit-2980_MixkitFree.ogg",
292         "breakdown"
293     );
294
295
296     //  4. load tracks
297     assets_manager_ptr->loadTrack(
298         "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
299         "Tree Star Moon - Dobranoc"
300     );
301
302     assets_manager_ptr->loadTrack(
303         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
304         "Tree Star Moon - Lighthouse"
305     );
306
307     assets_manager_ptr->loadTrack(
308         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
309         "Tree Star Moon - Sky Farm"
310     );
311
312     return;
313 }   /* loadAssets() */
```

5.28.2.3 main()

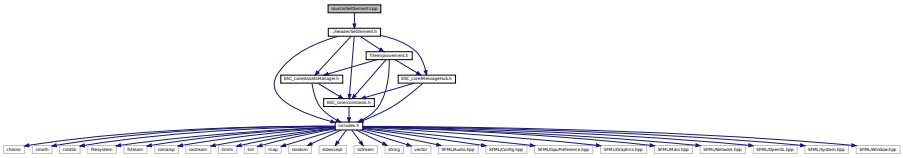
```
int main (
             int argc,
             char ** argv )
345 {
346     //  1. load assets
347     AssetsManager assets_manager;
348     loadAssets(&assets_manager);
349
350     //  2. construct render window
351     sf::RenderWindow* render_window_ptr = constructRenderWindow();
352
353     //  3. start game loop
354     bool quit_game = false;
355     assets_manager.playTrack();
356
357     while (not quit_game) {
358         Game game(render_window_ptr, &assets_manager);
359         quit_game = game.run();
360     }
361
```

```
362     //  4. clean up
363     render_window_ptr->close();
364     delete render_window_ptr;
365
366     return 0;
367 }   /* main() */
```

5.29 source/Settlement.cpp File Reference

Implementation file for the Settlement class.

`#include "../header/Settlement.h"`
Include dependency graph for Settlement.cpp:



5.29.1 Detailed Description

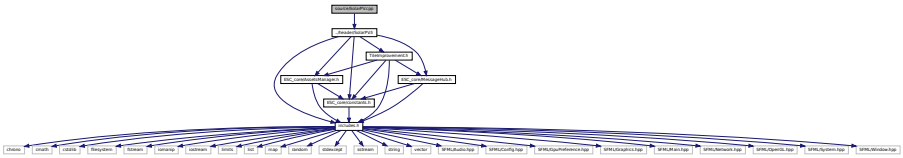
Implementation file for the Settlement class.

A base class for the tile improvement hierarchy.

5.30 source/SolarPV.cpp File Reference

Implementation file for the SolarPV class.

`#include "../header/SolarPV.h"`
Include dependency graph for SolarPV.cpp:



5.30.1 Detailed Description

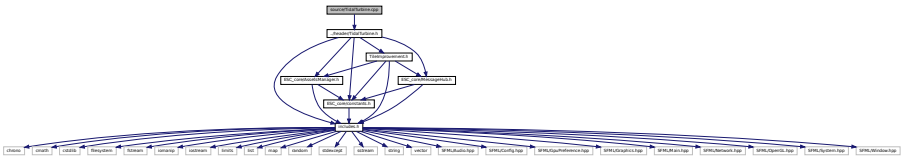
Implementation file for the SolarPV class.

A base class for the tile improvement hierarchy.

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

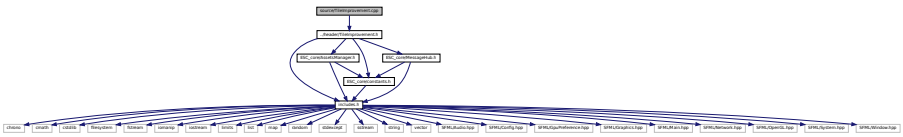
Implementation file for the TidalTurbine class.

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

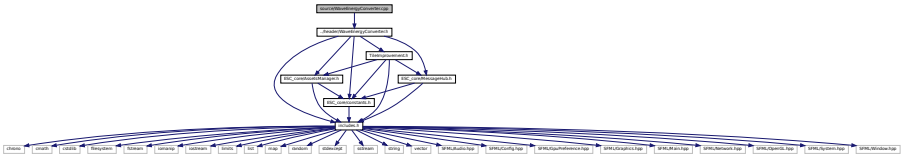
Implementation file for the TileImprovement class.

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.

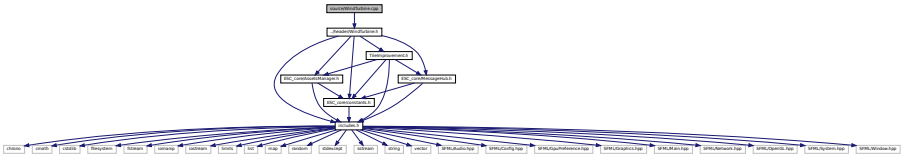
A base class for the tile improvement hierarchy.

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/. 276

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 275

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 40, 53, 105, 158, 168, 184, 203, 223, 240

Index