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:

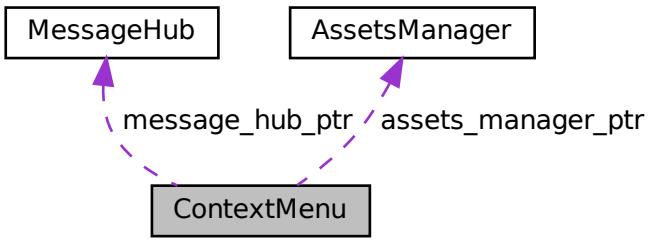
- header/ESC_core/AssetsManager.h
- source/ESC_core/AssetsManager.cpp

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               += "[ENTER]:  END TURN                 \n";
529             this->console_string               += "                                  \n";
530             this->console_string               += "[R]:  RESTART                      \n";
```

```
531             this->console_string                += "                                   \n";
532             this->console_string                += "[TAB]:  TOGGLE RESOURCE OVERLAY \n";
533             this->console_string                += "[T]:  TUTORIAL                  \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:

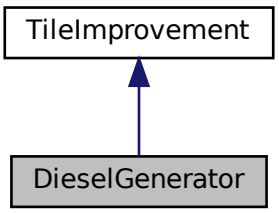
- header/ContextMenu.h
- source/ContextMenu.cpp

4.3 DieselGenerator Class Reference

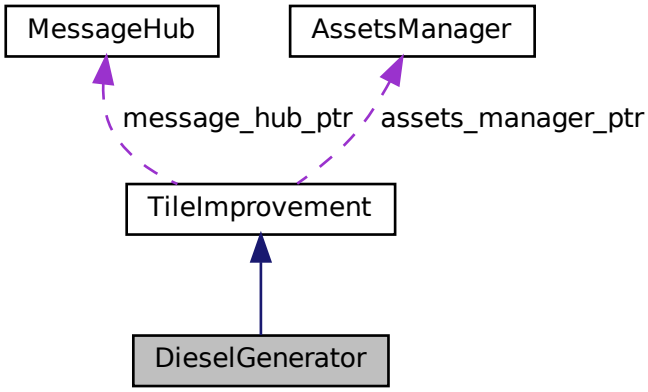
A settlement class (child class of TileImprovement).

```
#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 __repair (void)

  Helper method to repair the diesel generator.
- 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.

```
502   :
503 TileImprovement(
504     position_x,
505     position_y,
506     tile_resource,
507     event_ptr,
508     render_window_ptr,
509     assets_manager_ptr,
510     message_hub_ptr
511 )
512 {
513     //  1. set attributes
514
515     //  1.1. private
516     //...
517
518     //  1.2. public
519     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
520
521     this->is_running = false;
522
523     this->health = 100;
524
525     this->capacity_kW = 100;
526     this->upgrade_level = 1;
527
528     this->production_MWh = 0;
529     this->max_production_MWh = 72;
530
531     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
532     this->smoke_dx = 5 * SECONDS_PER_FRAME;
533     this->smoke_dy = -10 * SECONDS_PER_FRAME;
534     this->smoke_prob = 16 * SECONDS_PER_FRAME;
535
536     this->smoke_sprite_list = {};
537
538     this->fuel_cost = 0;
539     this->emissions_tonnes_CO2e = 0;
540
541     this->tile_improvement_string = "DIESEL GEN";
542
543     this->__setUpTileImprovementSpriteAnimated();
544
545     std::cout « "DieselGenerator constructed at " « this « std::endl;
546
547     return;
```

```
548 }   /* DieselGenerator() */
```

4.3.2.2 ~DieselGenerator()

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

Destructor for the DieselGenerator class.

```
921 {
922     std::cout « "DieselGenerator at " « this « " destroyed" « std::endl;
923
924     return;
925 }   /* ~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.

```
321 {
322     if (this->just_built) {
323         return;
324     }
325 
```

```
326
327     switch (this->event_ptr->key.code) {
328         case (sf::Keyboard::U): {
329             this->__upgrade();
330
331             break;
332         }
333
334
335         case (sf::Keyboard::W): {
336             if (this->production_menu_open) {
337                 this->production_MWh++;
338
339                 if (this->production_MWh > this->max_production_MWh) {
340                     this->production_MWh = 0;
341                 }
342
343                 this->__computeProductionCosts();
344                 this->assets_manager_ptr->getSound("interface click")->play();
345             }
346
347             break;
348         }
349
350
351         case (sf::Keyboard::S): {
352             if (this->production_menu_open) {
353                 this->production_MWh--;
354
355                 if (this->production_MWh < 0) {
356                     this->production_MWh = this->max_production_MWh;
357                 }
358
359                 this->__computeProductionCosts();
360                 this->assets_manager_ptr->getSound("interface click")->play();
361             }
362
363             break;
364         }
365
366
367         default: {
368             // do nothing!
369
370             break;
371         }
372     }
373
374
375     return;
376 }   /* __handleKeyPressEvents() */
```

4.3.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
391 {
392     if (this->just_built) {
393         return;
394     }
395
396     switch (this->event_ptr->mouseButton.button) {
397         case (sf::Mouse::Left): {
398             //...
399
400             break;
401         }
402
403
404         case (sf::Mouse::Right): {
405             //...
406
407             break;
408         }
409
410
```

```
411         default: {
412             // do nothing!
413 
414             break;
415         }
416     }
417 
418     return;
419 }   /* __handleMouseButtonEvents() */
```

4.3.3.6 __repair()

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

Helper method to repair the diesel generator.

Reimplemented from TileImprovement.

```
288 {
289     if (this->credits < DIESEL_GENERATOR_BUILD_COST) {
290         std::cout « "Cannot repair diesel generator: insufficient credits (need "
291             « DIESEL_GENERATOR_BUILD_COST « " K)" « std::endl;
292 
293         this->__sendInsufficientCreditsMessage();
294         return;
295     }
296 
297     TileImprovement :: __repair();
298 
299     this->just_upgraded = true;
300 
301     this->__sendCreditsSpentMessage(DIESEL_GENERATOR_BUILD_COST);
302     this->__sendTileStateRequest();
303     this->__sendGameStateRequest();
304 
305     return;
306 }   /* __repair() */
```

4.3.3.7 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
434 {
435     Message improvement_state_message;
436 
437     improvement_state_message.channel = GAME_CHANNEL;
438     improvement_state_message.subject = "improvement state";
439 
440     improvement_state_message.int_payload["dispatch_MWh"] = this->production_MWh;
441     improvement_state_message.int_payload["fuel_cost"] = this->fuel_cost;
442     improvement_state_message.int_payload["operation_maintenance_cost"] =
443         this->operation_maintenance_cost;
444     improvement_state_message.int_payload["emissions_tonnes_CO2e"] =
445         this->emissions_tonnes_CO2e;
446 
447     this->message_hub_ptr->sendMessage(improvement_state_message);
448 
449     std::cout « "Improvement state message sent by " « this « std::endl;
450 
451     return;
452 }   /* __sendImprovementStateMessage() */
```

4.3.3.8 __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() */
```

4.3.3.9 __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     TileImprovement :: __repair();
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() */
```

4.3.3.10 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
658 {
659     //  1. send improvement state message
660     this->__sendImprovementStateMessage();
661 
662     //  2. handle start/stop
663     if ((not this->is_running) and (this->production_MWh > 0)) {
664         this->is_running = true;
665         this->assets_manager_ptr->getSound("diesel start")->play();
666     }
667 
668     else if (this->is_running and (this->production_MWh <= 0)) {
669         this->is_running = false;
670         this->tile_improvement_sprite_animated[1].setScale(sf::Vector2f(1, 1));
671     }
672 
673     //  3. handle equipment health
674     if (this->is_running) {
675         this->health--;
676 
677         if (this->health <= 0) {
678             this->__breakdown();
679         }
680     }
681 
682     //  4. send tile state request (if selected)
683     if (this->is_selected) {
684         this->__sendTileStateRequest();
685     }
686 
687     return;
688 }   /* advanceTurn() */
```

4.3.3.11 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.

```
752 {
753     //  1. if just built, call base method and return
754     if (this->just_built) {
755         TileImprovement :: draw();
756 
757         return;
758     }
759 
760     //  2. handle upgrade effects
761     if (this->just_upgraded) {
762         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
763             this->tile_improvement_sprite_animated[i].setColor(
764                 sf::Color(
765                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
766                     255,
767                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
768                     255
769                 )
770             );
771 
772             this->tile_improvement_sprite_animated[i].setScale(
773                 sf::Vector2f(
774                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
775                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
776                 )
```

```
777             );
778         }
779 
780         this->upgrade_frame++;
781     }
782 
783     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
784         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
785             this->tile_improvement_sprite_animated[i].setColor(
786                 sf::Color(255,255,255,255)
787             );
788 
789             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
790         }
791 
792         this->just_upgraded = false;
793         this->upgrade_frame = 0;
794     }
795 
796 
797     //  3. draw first element of animated sprite
798     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
799 
800 
801     //  4. draw second element of animated sprite
802     double move_x = 0;
803     double move_y = 0;
804 
805     if (this->is_running) {
806         this->tile_improvement_sprite_animated[1].setScale(
807             sf::Vector2f(
808                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2),
809                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2)
810             )
811         );
812 
813         move_x = 1 * ((double)rand() / RAND_MAX) - 0.5;
814         move_y = 1 * ((double)rand() / RAND_MAX) - 0.5;
815 
816         this->tile_improvement_sprite_animated[1].move(move_x, move_y);
817     }
818 
819     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
820 
821     if (this->is_running) {
822         this->tile_improvement_sprite_animated[1].move(-1 * move_x, -1 * move_y);
823     }
824 
825 
826     //  5. draw smoke effects
827     if (this->is_running) {
828         if ((double)rand() / RAND_MAX < smoke_prob) {
829             this->smoke_sprite_list.push_back(
830                 sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
831             );
832 
833             this->smoke_sprite_list.back().setOrigin(
834                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
835                 this->smoke_sprite_list.back().getLocalBounds().height / 2
836             );
837 
838             this->smoke_sprite_list.back().setPosition(
839                 this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
840                 this->position_y - 33
841             );
842         }
843     }
844 
845     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
846 
847     double alpha = 255;
848 
849     while (iter != this->smoke_sprite_list.end()) {
850         this->render_window_ptr->draw(*iter);
851 
852         alpha = (*iter).getColor().a;
853 
854         alpha -= this->smoke_da;
855 
856         if (alpha <= 0) {
857             iter = this->smoke_sprite_list.erase(iter);
858             continue;
859         }
860 
861         (*iter).setColor(sf::Color(255, 255, 255, alpha));
862 
863         (*iter).move(
```

```
864             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
865             this->smoke_dy
866         );
867 
868         (*iter).rotate(((double)rand() / RAND_MAX));
869 
870         iter++;
871     }
872 
873 
874     //  6. handle dispatch illustration
875     if (this->production_MWh > 0) {
876         this->dispatch_text.setString(std::to_string(this->production_MWh));
877         this->__drawDispatch();
878     }
879 
880 
881     //  7. draw production menu
882     if (this->production_menu_open) {
883         this->render_window_ptr->draw(this->production_menu_backing);
884         this->render_window_ptr->draw(this->production_menu_backing_text);
885 
886         this->__drawProductionMenu();
887     }
888 
889 
890     //  8. handle broken effects
891     if (this->is_broken) {
892         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
893             this->tile_improvement_sprite_animated[i].setColor(
894                 sf::Color(
895                     255,
896                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
897                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
898                     255
899                 )
900             );
901         }
902     }
903 
904     this->frame++;
905     return;
906 }   /* draw() */
```

4.3.3.12 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
565 {
566     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
567 
568     //                  32 char x 17 line console "--------------------------------\n";
569     std::string options_substring                = "CAPACITY:    ";
570     options_substring                           += std::to_string(this->capacity_kW);
571     options_substring                           += " kW (level ";
572     options_substring                           += std::to_string(this->upgrade_level);
573     options_substring                           += ")\n";
574 
575     options_substring                           += "PRODUCTION:  ";
576     options_substring                           += std::to_string(this->production_MWh);
577     options_substring                           += " MWh (MAX ";
578     options_substring                           += std::to_string(this->max_production_MWh);
579     options_substring                           += ")\n";
580 
581     options_substring                           += "HEALTH:      ";
582     options_substring                           += std::to_string(this->health);
```

```
583     options_substring                           += "/100";
584
585     if (this->health <= 0) {
586         options_substring                       += " ** BROKEN! **\n";
587     }
588
589     else {
590         options_substring                       += "\n";
591     }
592
593     options_substring                           += "                                  \n";
594     options_substring                           += "  **** DIESEL GEN OPTIONS ****   \n";
595     options_substring                           += "                                  \n";
596
597     if (this->is_broken) {
598         options_substring                       += "     [R]:  REPAIR (";
599         options_substring                       += std::to_string(DIESEL_GENERATOR_BUILD_COST);
600         options_substring                       += " K)\n";
601     }
602
603     else {
604         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
605     }
606
607     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
608         options_substring                           += "     [U]:  + 100 kW (";
609         options_substring                           += std::to_string(upgrade_cost);
610         options_substring                           +=" K)\n";
611     }
612
613     options_substring                           += "HOLD [P]:  SCRAP (";
614     options_substring                           += std::to_string(SCRAP_COST);
615     options_substring                           += " K)";
616
617     return options_substring;
618 }   /* getTileOptionsSubstring() */
```

4.3.3.13 processEvent()

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

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

Reimplemented from TileImprovement.

```
703 {
704     TileImprovement :: processEvent();
705
706     if (this->event_ptr->type == sf::Event::KeyPressed) {
707         this->__handleKeyPressEvents();
708     }
709
710     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
711         this->__handleMouseButtonEvents();
712     }
713
714     return;
715 }   /* processEvent() */
```

4.3.3.14 processMessage()

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

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

Reimplemented from TileImprovement.

```
730 {
731     TileImprovement :: processMessage();
732
733     //...
734
735     return;
736 }   /* processMessage() */
```

4.3.3.15 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.

```
635 {
636     TileImprovement :: setIsSelected(is_selected);
637 
638     if (this->is_running and this->is_selected) {
639         this->assets_manager_ptr->getSound("diesel running")->play();
640     }
641 
642     return;
643 }   /* 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:

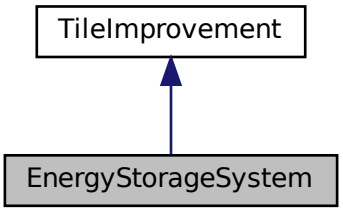
- header/DieselGenerator.h
- source/DieselGenerator.cpp

4.4 EnergyStorageSystem Class Reference

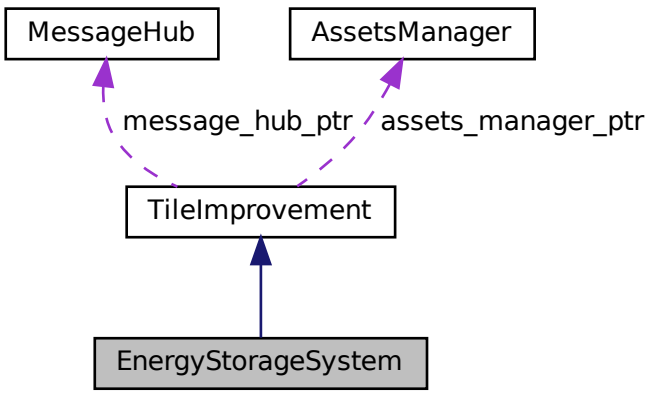
A settlement class (child class of TileImprovement).

```
#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:

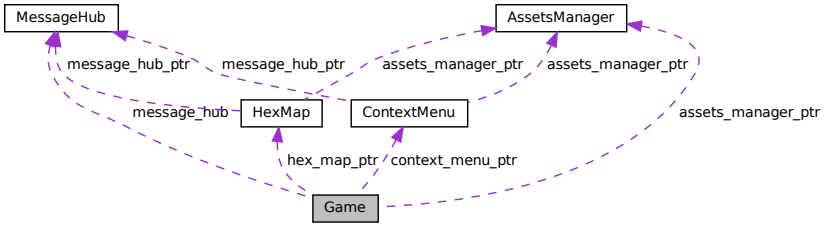
- header/EnergyStorageSystem.h
- source/EnergyStorageSystem.cpp

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.
- bool show_tutorial

  A boolean indicating whether or not to show the tutorial.
- bool turn_end

  A boolean indicating a turn end.
- unsigned long long int frame

  The current frame of the game.
- double time_since_start_s

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

  Current game year.
- int month

  Current game month.
- int population

  Current population.
- int credits

  Current balance of credits.
- int demand_MWh

  Current energy demand [MWh].
- int cumulative_emissions_tonnes

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

  The demand in the previous turn.
- int demand_served_MWh

  The demand served at the end of a turn.
- int demand_remaining_MWh

  The demand remaining at the end of a turn.
- int overproduction_MWh

  The amount of overproduction at the end of a turn.
- int turn_fuel_cost

  The cost of fuel at the end of a turn.
- int turn_operation_maintenance_cost

  The cost of operation and maintenance at the end of a turn.
- int turn_emissions_tonnes

  The amount of emissions at the end of a turn.
- int dispatch_income

  The amount earned from dispatch at the end of a turn.
- int overproduction_penalty

The penalty for overproduction.

- int net_credit_flow

  The net credit flow at the end of a turn.
- int consecutive_zero_emissions_months

  The number of recent, consecutive zero emission months.
- size_t substring_idx

  The index of the turn summary substring.
- std::string turn_summary_string

  A string representation of the end of turn summary.
- sf::Text turn_summary_text

  A text representation (drawable) of the end of turn summary.
- 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 __updatePopulation (void)

  Helper method to update (i.e. grow) population.
- 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 __summarizeTurn (void)

  Helper method to generate end of turn summary.
- void __drawLossDemand (void)

  Helper method to draw loss (demand) pop-up.
- void __drawLossCredits (void)

  Helper method to draw loss (credits) pop-up.
- void __drawLossEmissions (void)

  Helper method to draw loss (emissions) pop-up.
- void __drawVictory (void)

  Helper method to draw victory pop-up.
- void __drawTurnSummary (void)

  Helper method to draw turn summary.
- 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.

```
1473 {
1474     //  1. set attributes
1475
1476     //  1.1. private
1477     this->render_window_ptr = render_window_ptr;
1478
1479     this->assets_manager_ptr = assets_manager_ptr;
1480
1481     //  1.2. public
1482     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
1483
1484     this->quit_game = false;
1485     this->game_loop_broken = false;
1486     this->show_frame_clock_overlay = false;
1487     this->check_terminating_conditions = false;
1488     this->show_tutorial = false;
1489     this->turn_end = false;
1490
1491     this->frame = 0;
1492     this->time_since_start_s = 0;
1493
1494     this->message_deadlock = false;
1495     this->message_deadlock_frame = 0;
1496
1497     double seconds_since_epoch = time(NULL);
1498     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
1499
1500     this->year = 1970 + (int)years_since_epoch;
1501     this->month = (years_since_epoch - (int)years_since_epoch) * 12 + 1;
1502     while (this->month > 12) {
1503         this->month -= 12;
1504     }
1505
1506     this->population = 0;
1507     this->credits = STARTING_CREDITS;
1508     this->demand_MWh = 0;
1509     this->cumulative_emissions_tonnes = 0;
1510
1511     this->past_demand_MWh = 0;
1512
1513     this->demand_vec_MWh.resize(30, 0);
1514
1515     this->demand_served_MWh = 0;
1516     this->demand_remaining_MWh = 0;
1517     this->overproduction_MWh = 0;
1518     this->turn_fuel_cost = 0;
1519     this->turn_operation_maintenance_cost = 0;
1520     this->turn_emissions_tonnes = 0;
1521
1522     this->overproduction_penalty = 0;
1523     this->dispatch_income = 0;
1524     this->net_credit_flow = 0;
1525
1526     this->consecutive_zero_emissions_months = 0;
1527
1528     this->substring_idx = 0;
1529     this->turn_summary_string = "";
1530
1531     this->turn_summary_text.setFont(
1532         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
1533     );
1534     this->turn_summary_text.setCharacterSize(16);
1535     this->turn_summary_text.setFillColor(MONOCHROME_TEXT_GREEN);
1536     this->turn_summary_text.setPosition(GAME_WIDTH - 400 + 64, 64);
1537
1538     this->hex_map_ptr = new HexMap(
1539         6,
1540         &(this->event),
1541         this->render_window_ptr,
1542         this->assets_manager_ptr,
1543         &(this->message_hub)
1544     );
1545
1546     this->context_menu_ptr = new ContextMenu(
1547         &(this->event),
1548         this->render_window_ptr,
1549         this->assets_manager_ptr,
```

```
1550         &(this->message_hub)
1551     );
1552
1553     //  2. add message channel(s)
1554     this->message_hub.addChannel(GAME_CHANNEL);
1555     this->message_hub.addChannel(GAME_STATE_CHANNEL);
1556
1557     this->__sendGameStateMessage();
1558
1559     std::cout << "Game constructed at " << this << std::endl;
1560
1561     return;
1562 }   /* Game() */
```

4.5.2.2 ~Game()

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

Destructor for the Game class.

```
1688 {
1689     //  1. clean up attributes
1690     delete this->hex_map_ptr;
1691     delete this->context_menu_ptr;
1692
1693     std::cout << "Game at " << this << " destroyed" << std::endl;
1694
1695     return;
1696 }   /* ~Game() */
```

4.5.3 Member Function Documentation

4.5.3.1 __advanceTurn()

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

Helper method to advance turn.

```
172 {
173     //  1. advance turn, raise turn end flag
174     this->turn++;
175     this->turn_end = true;
176
177     //  2. reset turn summary attributes
178     this->demand_served_MWh = 0;
179     this->demand_remaining_MWh = 0;
180     this->overproduction_MWh = 0;
181     this->turn_fuel_cost = 0;
182     this->turn_operation_maintenance_cost = 0;
183     this->turn_emissions_tonnes = 0;
184
185     this->overproduction_penalty = 0;
186     this->dispatch_income = 0;
187     this->net_credit_flow = 0;
188
189     //  3. advance month/year
190     this->month++;
191     if (this->month > 12) {
192         this->year++;
193         this->month = 1;
194     }
195
196     //  4. update population
197     if (this->turn == 1) {
198         this->population = STARTING_POPULATION;
```

```
199     }
200
201     else {
202         this->__updatePopulation();
203     }
204
205     //  5. update demand
206     this->__computeCurrentDemand();
207
208     //  6. send turn advance message
209     this->__sendTurnAdvanceMessage();
210     this->__sendGameStateMessage();
211
212 }   /* __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     //  1. loss emissions
98     if (this->cumulative_emissions_tonnes >= EMISSIONS_LIFETIME_LIMIT_TONNES) {
99         this->assets_manager_ptr->getSound("loss")->play();
100         this->game_phase = GamePhase :: LOSS_EMISSIONS;
101     }
102
103     //  2. loss demand
104     else if (this->demand_remaining_MWh > 0) {
105         this->assets_manager_ptr->getSound("loss")->play();
106         this->game_phase = GamePhase :: LOSS_DEMAND;
107     }
108
109     //  3. loss credits
110     else if (this->credits < 0) {
111         this->assets_manager_ptr->getSound("loss")->play();
112         this->game_phase = GamePhase :: LOSS_CREDITS;
113     }
114
115     //  4. victory
116     else if (
117         (this->population >= 1000) and
118         (this->consecutive_zero_emissions_months >= 12)
119     ) {
120         this->assets_manager_ptr->getSound("victory")->play();
121         this->game_phase = GamePhase :: VICTORY;
122     }
123
124     //  5. send game state message
125     //this->__sendGameStateMessage();
126
127     return;
128 }   /* __checkTerminatingConditions() */
```

4.5.3.3 __computeCurrentDemand()

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

Helper method to compute current energy demand.

```
227 {
228     this->past_demand_MWh = this->demand_MWh;
229
230     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
231     std::default_random_engine generator(seed);
232
```

```
233     std::normal_distribution<double> normal_dist(
234         MEAN_DAILY_DEMAND_RATIOS[this->month - 1],
235         STDEV_DAILY_DEMAND_RATIOS[this->month - 1]
236     );
237 
238     double demand_MWh = 0;
239 
240     for (int i = 0; i < 30; i++) {
241         this->demand_vec_MWh[i] =
242             normal_dist(generator) * MAXIMUM_DAILY_DEMAND_PER_CAPITA * this->population;
243 
244         demand_MWh += this->demand_vec_MWh[i];
245     }
246 
247     this->demand_MWh = round(demand_MWh);
248 
249     return;
250 }   /* __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.

```
1396 {
1397     this->__drawHUD();
1398 
1399     if (this->show_frame_clock_overlay) {
1400         this->__drawFrameClockOverlay();
1401     }
1402 
1403     if (this->show_tutorial) {
1404 
1405     }
1406 
1407     else if (not this->turn_summary_string.empty()) {
1408         this->__drawTurnSummary();
1409     }
1410 
1411     switch (this->game_phase) {
1412         case (GamePhase :: LOSS_DEMAND): {
1413             this->__drawLossDemand();
1414 
1415             break;
1416         }
1417 
1418 
1419         case (GamePhase :: LOSS_CREDITS): {
1420             this->__drawLossCredits();
1421 
1422             break;
1423         }
1424 
1425 
1426         case (GamePhase :: LOSS_EMISSIONS): {
1427             this->__drawLossEmissions();
1428 
1429             break;
1430         }
1431 
1432 
1433         case (GamePhase ::VICTORY): {
1434             this->__drawVictory();
1435 
1436             break;
1437         }
1438 
1439 
1440         default: {
1441             // do nothing!
1442 
1443             break;
1444         }
1445     }
1446 
1447     return;
1448 }   /* draw() */
```

4.5.3.5 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
1219 {
1220     std::string frame_clock_string = "FRAME: ";
1221     frame_clock_string += std::to_string(this->frame);
1222     frame_clock_string += "\nTIME SINCE START [s]: ";
1223     frame_clock_string += std::to_string(this->time_since_start_s);
1224 
1225     sf::Text frame_clock_text(
1226         frame_clock_string,
1227         *(this->assets_manager_ptr->getFont("DroidSansMono")),
1228         16
1229     );
1230 
1231     sf::RectangleShape frame_clock_backing(
1232         sf::Vector2f(
1233             1.02 * frame_clock_text.getLocalBounds().width,
1234             1.20 * frame_clock_text.getLocalBounds().height
1235         )
1236     );
1237     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
1238 
1239     this->render_window_ptr->draw(frame_clock_backing);
1240     this->render_window_ptr->draw(frame_clock_text);
1241 
1242     return;
1243 }   /* __drawFrameClockOverlay() */
```

4.5.3.6 __drawHUD()

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

Helper method to heads-up display (HUD).

```
1258 {
1259     //  1. first line (top)
1260     std::string HUD_string = "YEAR: ";
1261     HUD_string += std::to_string(this->year);
1262 
1263     HUD_string += "    MONTH: ";
1264     HUD_string += std::to_string(this->month);
1265 
1266     HUD_string += "    POPULATION: ";
1267     HUD_string += std::to_string(this->population);
1268 
1269     HUD_string += "    CREDITS: ";
1270     HUD_string += std::to_string(this->credits);
1271     HUD_string += " K";
1272 
1273     HUD_string += "    CURRENT DEMAND: ";
1274     HUD_string += std::to_string(this->demand_MWh);
1275     HUD_string += " MWh";
1276 
1277     sf::Text HUD_text(
1278         HUD_string,
1279         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1280         16
1281     );
1282 
1283     HUD_text.setPosition(
1284         (800 - HUD_text.getLocalBounds().width) / 2,
1285         8
1286     );
1287 
1288     HUD_text.setFillColor(MONOCHROME_TEXT_GREEN);
1289 
1290     this->render_window_ptr->draw(HUD_text);
1291 
1292 
1293     //  2. second line (top)
1294     HUD_string = "CUMULATIVE EMISSIONS: ";
```

```
1295     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
1296     HUD_string += " tonnes (CO2e)";
1297 
1298     HUD_string += "    LIFETIME LIMIT: ";
1299     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
1300     HUD_string += " tonnes (CO2e)";
1301 
1302     HUD_text.setString(HUD_string);
1303 
1304     HUD_text.setPosition(
1305         (800 - HUD_text.getLocalBounds().width) / 2,
1306         35
1307     );
1308 
1309     this->render_window_ptr->draw(HUD_text);
1310 
1311 
1312     //  3. third line (bottom)
1313     HUD_string = "GAME PHASE: ";
1314 
1315     switch (this->game_phase) {
1316         case (GamePhase :: BUILD_SETTLEMENT): {
1317             HUD_string += "BUILD SETTLEMENT";
1318 
1319             break;
1320         }
1321 
1322 
1323         case (GamePhase :: SYSTEM_MANAGEMENT): {
1324             HUD_string += "SYSTEM MANAGEMENT";
1325 
1326             break;
1327         }
1328 
1329 
1330         case (GamePhase :: LOSS_EMISSIONS): {
1331             HUD_string += "LOSS (EMISSIONS)";
1332 
1333             break;
1334         }
1335 
1336 
1337         case (GamePhase :: LOSS_DEMAND): {
1338             HUD_string += "LOSS (DEMAND)";
1339 
1340             break;
1341         }
1342 
1343 
1344         case (GamePhase :: LOSS_CREDITS): {
1345             HUD_string += "LOSS (CREDITS)";
1346 
1347             break;
1348         }
1349 
1350 
1351         case (GamePhase :: VICTORY): {
1352             HUD_string += "VICTORY";
1353 
1354             break;
1355         }
1356 
1357 
1358         default: {
1359             HUD_string += "???";
1360 
1361             break;
1362         }
1363     }
1364 
1365     HUD_string += "    TURN: ";
1366     HUD_string += std::to_string(this->turn);
1367 
1368     HUD_string += "    CONSECUTIVE ZERO EMISSIONS MONTHS: ";
1369     HUD_string += std::to_string(this->consecutive_zero_emissions_months);
1370 
1371     HUD_text.setString(HUD_string);
1372 
1373     HUD_text.setPosition(
1374         (800 - HUD_text.getLocalBounds().width) / 2,
1375         GAME_HEIGHT - 35
1376     );
1377 
1378     this->render_window_ptr->draw(HUD_text);
1379 
1380     return;
1381 }    /* __drawHUD() */
```

4.5.3.7 __drawLossCredits()

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

Helper method to draw loss (credits) pop-up.

```
989 {
990     //  1. construct loss text and backing rectangle
991     std::string loss_credits_string  = "    LOSS! - RAN OUT OF CREDITS     \n";
992     loss_credits_string             += "     press any key to restart      ";
993 
994     sf::Text loss_credits_text(
995         loss_credits_string,
996         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
997         32
998     );
999 
1000     loss_credits_text.setOrigin(
1001         loss_credits_text.getLocalBounds().width / 2,
1002         loss_credits_text.getLocalBounds().height / 2
1003     );
1004 
1005     loss_credits_text.setPosition(400, GAME_HEIGHT / 2);
1006 
1007     sf::RectangleShape backing_rectangle(
1008         sf::Vector2f(
1009             1.1 * loss_credits_text.getLocalBounds().width,
1010             1.5 * loss_credits_text.getLocalBounds().height
1011         )
1012     );
1013 
1014     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1015 
1016     backing_rectangle.setOrigin(
1017         backing_rectangle.getLocalBounds().width / 2,
1018         backing_rectangle.getLocalBounds().height / 2
1019     );
1020 
1021     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1022 
1023     //  3. colour cycle and draw
1024     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1025         loss_credits_text.setFillColor(MONOCHROME_TEXT_RED);
1026     }
1027 
1028     else {
1029         loss_credits_text.setFillColor(sf::Color(255, 255, 255, 255));
1030     }
1031 
1032     this->render_window_ptr->draw(backing_rectangle);
1033     this->render_window_ptr->draw(loss_credits_text);
1034 
1035     return;
1036 }   /* __drawLossCredits() */
```

4.5.3.8 __drawLossDemand()

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

Helper method to draw loss (demand) pop-up.

```
927 {
928     //  1. construct alarm text and backing rectangle
929     std::string loss_demand_string  = "   LOSS! - FAILED TO MEET DEMAND   \n";
930     loss_demand_string             += "     press any key to restart      ";
931 
932     sf::Text loss_demand_text(
933         loss_demand_string,
934         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
935         32
```

```
936     );
937
938     loss_demand_text.setOrigin(
939         loss_demand_text.getLocalBounds().width / 2,
940         loss_demand_text.getLocalBounds().height / 2
941     );
942
943     loss_demand_text.setPosition(400, GAME_HEIGHT / 2);
944
945     sf::RectangleShape backing_rectangle(
946         sf::Vector2f(
947             1.1 * loss_demand_text.getLocalBounds().width,
948             1.5 * loss_demand_text.getLocalBounds().height
949         )
950     );
951
952     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
953
954     backing_rectangle.setOrigin(
955         backing_rectangle.getLocalBounds().width / 2,
956         backing_rectangle.getLocalBounds().height / 2
957     );
958
959     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
960
961     //  3. colour cycle and draw
962     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
963         loss_demand_text.setFillColor(MONOCHROME_TEXT_RED);
964     }
965
966     else {
967         loss_demand_text.setFillColor(sf::Color(255, 255, 255, 255));
968     }
969
970     this->render_window_ptr->draw(backing_rectangle);
971     this->render_window_ptr->draw(loss_demand_text);
972
973     return;
974 }   /* __drawLossDemand() */
```

4.5.3.9 __drawLossEmissions()

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

Helper method to draw loss (emissions) pop-up.

```
1051 {
1052     //  1. construct loss text and backing rectangle
1053     std::string loss_emissions_string  = "    LOSS! - EXCESSIVE EMISSIONS    \n";
1054     loss_emissions_string             += "     press any key to restart      ";
1055
1056     sf::Text loss_emissions_text(
1057         loss_emissions_string,
1058         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1059         32
1060     );
1061
1062     loss_emissions_text.setOrigin(
1063         loss_emissions_text.getLocalBounds().width / 2,
1064         loss_emissions_text.getLocalBounds().height / 2
1065     );
1066
1067     loss_emissions_text.setPosition(400, GAME_HEIGHT / 2);
1068
1069     sf::RectangleShape backing_rectangle(
1070         sf::Vector2f(
1071             1.1 * loss_emissions_text.getLocalBounds().width,
1072             1.5 * loss_emissions_text.getLocalBounds().height
1073         )
1074     );
1075
1076     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1077
1078     backing_rectangle.setOrigin(
1079         backing_rectangle.getLocalBounds().width / 2,
1080         backing_rectangle.getLocalBounds().height / 2
1081     );
1082
```

```
1083     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1084
1085     //  3. colour cycle and draw
1086     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1087         loss_emissions_text.setFillColor(MONOCHROME_TEXT_RED);
1088     }
1089
1090     else {
1091         loss_emissions_text.setFillColor(sf::Color(255, 255, 255, 255));
1092     }
1093
1094     this->render_window_ptr->draw(backing_rectangle);
1095     this->render_window_ptr->draw(loss_emissions_text);
1096
1097     return;
1098 }   /* __drawLossEmissions() */
```

4.5.3.10 __drawTurnSummary()

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

Helper method to draw turn summary.

```
1175 {
1176     if (this->substring_idx < this->turn_summary_string.size()) {
1177         this->assets_manager_ptr->getSound("console string print")->play();
1178
1179         this->turn_summary_text.setString(
1180             this->turn_summary_string.substr(0, this->substring_idx)
1181         );
1182
1183         while (
1184             (this->turn_summary_string.substr(0, this->substring_idx).back() == ' ') or
1185             (this->turn_summary_string.substr(0, this->substring_idx).back() == '\n')
1186         ) {
1187             this->substring_idx++;
1188
1189             if (this->substring_idx == this->turn_summary_string.size() - 1) {
1190                 this->turn_summary_text.setString(
1191                     this->turn_summary_string.substr(0, this->substring_idx)
1192                 );
1193
1194                 break;
1195             }
1196         }
1197
1198         this->substring_idx++;
1199     }
1200
1201     this->render_window_ptr->draw(this->turn_summary_text);
1202
1203     return;
1204 }   /* __drawTurnSummary() */
```

4.5.3.11 __drawVictory()

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

Helper method to draw victory pop-up.

```
1113 {
1114     //  1. construct victory text and backing rectangle
1115     std::string victory_string  = "        **** VICTORY! ****         \n";
1116     victory_string             += "     press any key to restart      ";
1117
1118     sf::Text victory_text(
1119         victory_string,
1120         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1121         32
```

```
1122     );
1123
1124     victory_text.setOrigin(
1125         victory_text.getLocalBounds().width / 2,
1126         victory_text.getLocalBounds().height / 2
1127     );
1128
1129     victory_text.setPosition(400, GAME_HEIGHT / 2);
1130
1131     sf::RectangleShape backing_rectangle(
1132         sf::Vector2f(
1133             1.1 * victory_text.getLocalBounds().width,
1134             1.5 * victory_text.getLocalBounds().height
1135         )
1136     );
1137
1138     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1139
1140     backing_rectangle.setOrigin(
1141         backing_rectangle.getLocalBounds().width / 2,
1142         backing_rectangle.getLocalBounds().height / 2
1143     );
1144
1145     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1146
1147     //  3. colour cycle and draw
1148     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1149         victory_text.setFillColor(MONOCHROME_TEXT_GREEN);
1150     }
1151
1152     else {
1153         victory_text.setFillColor(sf::Color(255, 255, 255, 255));
1154     }
1155
1156     this->render_window_ptr->draw(backing_rectangle);
1157     this->render_window_ptr->draw(victory_text);
1158
1159     return;
1160 }   /* __drawVictory() */
```

4.5.3.12 __handleImprovementStateMessage()

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

Helper method to handle improvement state messages.

```
352 {
353     //  1. dispatch
354     if (improvement_state_message.int_payload.count("dispatch_MWh") > 0) {
355         this->demand_served_MWh += improvement_state_message.int_payload["dispatch_MWh"];
356     }
357
358     //  2. fuel costs
359     if (improvement_state_message.int_payload.count("fuel_cost") > 0) {
360         this->turn_fuel_cost += improvement_state_message.int_payload["fuel_cost"];
361     }
362
363     //  3. operation and maintenance costs
364     if (improvement_state_message.int_payload.count("operation_maintenance_cost") > 0) {
365         this->turn_operation_maintenance_cost +=
366             improvement_state_message.int_payload["operation_maintenance_cost"];
367     }
368
369     //  4. emissions
370     if (improvement_state_message.int_payload.count("emissions_tonnes_CO2e") > 0) {
371         double emissions_tonnes_CO2e =
372             improvement_state_message.int_payload["emissions_tonnes_CO2e"];
373
374         this->cumulative_emissions_tonnes += emissions_tonnes_CO2e;
375         this->turn_emissions_tonnes += emissions_tonnes_CO2e;
376     }
377
378     return;
379 }   /* __handleImprovementStateMessage() */
```

4.5.3.13 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
265 {
266     switch (this->event.key.code) {
267         case (sf::Keyboard::Enter): {
268             if (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT) {
269                 this->__advanceTurn();
270             }
271
272             break;
273         }
274
275
276         case (sf::Keyboard::Tilde): {
277             this->__toggleFrameClockOverlay();
278
279             break;
280         }
281
282
283         case (sf::Keyboard::Tab): {
284             this->hex_map_ptr->toggleResourceOverlay();
285
286             break;
287         }
288
289
290         default: {
291             // do nothing!
292
293             break;
294         }
295     }
296
297     return;
298 }   /* __handleKeyPressEvents() */
```

4.5.3.14 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
313 {
314     switch (this->event.mouseButton.button) {
315         case (sf::Mouse::Left): {
316             //...
317
318             break;
319         }
320
321
322         case (sf::Mouse::Right): {
323             //...
324
325             break;
326         }
327
328
329         default: {
330             // do nothing!
331
332             break;
333         }
334     }
335
336     return;
337 }   /* __handleMouseButtonEvents() */
```

4.5.3.15 __insufficientCreditsAlarm()

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

Helper method to sound and display and insufficient credits alarm.

```
694 {
695     //  1. sound buzzer
696     this->assets_manager_ptr->getSound("insufficient credits")->play();
697 
698     //  2. construct alarm text and backing rectangle
699     sf::Text insufficient_credits_text(
700         "INSUFFICIENT CREDITS",
701         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
702         32
703     );
704 
705     insufficient_credits_text.setOrigin(
706         insufficient_credits_text.getLocalBounds().width / 2,
707         insufficient_credits_text.getLocalBounds().height / 2
708     );
709 
710     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
711 
712     sf::RectangleShape backing_rectangle(
713         sf::Vector2f(
714             1.1 * insufficient_credits_text.getLocalBounds().width,
715             1.5 * insufficient_credits_text.getLocalBounds().height
716         )
717     );
718 
719     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
720 
721     backing_rectangle.setOrigin(
722         backing_rectangle.getLocalBounds().width / 2,
723         backing_rectangle.getLocalBounds().height / 2
724     );
725 
726     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
727 
728     //  3. display loop (blocking ~3 seconds)
729     bool red_flag = true;
730     int alarm_frame = 0;
731     double time_since_alarm_s = 0;
732 
733     sf::Clock alarm_clock;
734 
735     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
736 
737 
738         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
739 
740         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
741             while (this->render_window_ptr->pollEvent(this->event)) {
742                 // do nothing!
743             }
744 
745             this->render_window_ptr->clear();
746 
747             this->hex_map_ptr->draw();
748             this->context_menu_ptr->draw();
749             this->__draw();
750 
751             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
752                 if (red_flag) {
753                     red_flag = false;
754                 }
755 
756                 else {
757                     red_flag = true;
758                 }
759             }
760 
761             if (red_flag) {
762                 insufficient_credits_text.setFillColor(MONOCHROME_TEXT_RED);
763             }
764 
765             else {
766                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
767             }
768 
769             this->render_window_ptr->draw(backing_rectangle);
770             this->render_window_ptr->draw(insufficient_credits_text);
771 
```

```
772             this->render_window_ptr->display();
773
774             alarm_frame++;
775             this->frame++;
776         }
777
778         // check track status, move to next if stopped
779         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
780             this->assets_manager_ptr->nextTrack();
781             this->assets_manager_ptr->playTrack();
782         }
783     }
784
785     return;
786 }   /* __insufficientCreditsAlarm( */
```

4.5.3.16 __processEvent()

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

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

```
394 {
395     if (this->event.type == sf::Event::Closed) {
396         this->quit_game = true;
397         this->game_loop_broken = true;
398     }
399
400     if (this->event.type == sf::Event::KeyPressed) {
401         this->__handleKeyPressEvents();
402     }
403
404     if (this->event.type == sf::Event::MouseButtonPressed) {
405         this->__handleMouseButtonEvents();
406     }
407
408     return;
409 }   /* __processEvent() */
```

4.5.3.17 __processMessage()

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

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

```
565 {
566     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
567         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
568
569         if (game_channel_message.subject == "quit game") {
570             this->quit_game = true;
571             this->game_loop_broken = true;
572
573             std::cout « "Quit game message received by " « this « std::endl;
574             this->message_hub.popMessage(GAME_CHANNEL);
575         }
576
577         if (game_channel_message.subject == "restart game") {
578             this->game_loop_broken = true;
579
580             std::cout « "Restart game message received by " « this « std::endl;
581             this->message_hub.popMessage(GAME_CHANNEL);
582         }
583
584         if (game_channel_message.subject == "state request") {
585             std::cout « "Game state request message received by " « this « std::endl;
586
587             this->__sendGameStateMessage();
588             this->message_hub.popMessage(GAME_CHANNEL);
```

```
589         }
590
591         if (game_channel_message.subject == "credits spent") {
592             this->credits -= game_channel_message.int_payload["credits spent"];
593
594             std::cout « "Credits spent message (" «
595                 game_channel_message.int_payload["credits spent"] « ") received by "
596                 « this « std::endl;
597
598             std::cout « "Current credits (Game): " « this->credits « " K" «
599                 std::endl;
600
601             this->message_hub.popMessage(GAME_CHANNEL);
602         }
603
604         if (game_channel_message.subject == "insufficient credits") {
605             std::cout « "Insufficient credits message received by " « this «
606                 std::endl;
607
608             this->__insufficientCreditsAlarm();
609
610             this->message_hub.popMessage(GAME_CHANNEL);
611         }
612
613         if (game_channel_message.subject == "update game phase") {
614             std::cout « "Update game phase message received by " « this « std::endl;
615
616             if (
617                 game_channel_message.string_payload["game phase"] == "system management"
618             ) {
619                 this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
620                 this->__advanceTurn();
621             }
622
623             else if (
624                 game_channel_message.string_payload["game phase"] == "loss emissions"
625             ) {
626                 this->game_phase = GamePhase :: LOSS_EMISSIONS;
627             }
628
629             else if (
630                 game_channel_message.string_payload["game phase"] == "loss demand"
631             ) {
632                 this->game_phase = GamePhase :: LOSS_DEMAND;
633             }
634
635             else if (
636                 game_channel_message.string_payload["game phase"] == "loss credits"
637             ) {
638                 this->game_phase = GamePhase :: LOSS_CREDITS;
639             }
640
641             else if (
642                 game_channel_message.string_payload["game phase"] == "victory"
643             ) {
644                 this->game_phase = GamePhase :: VICTORY;
645             }
646
647             this->message_hub.popMessage(GAME_CHANNEL);
648         }
649
650         if (game_channel_message.subject == "improvement state") {
651             std::cout « "Improvement state message received by " « this « std::endl;
652
653             this->__handleImprovementStateMessage(game_channel_message);
654
655             this->message_hub.popMessage(GAME_CHANNEL);
656         }
657     }
658
659     if (not this->message_hub.isEmpty(GAME_STATE_CHANNEL)) {
660         Message game_state_message =
661             this->message_hub.receiveMessage(GAME_STATE_CHANNEL);
662
663         if (game_state_message.subject == "turn advance") {
664             if (game_state_message.number_of_reads > 0) {
665                 std::cout « "Turn advance message received by " « this « std::endl;
666                 this->message_hub.popMessage(GAME_STATE_CHANNEL);
667             }
668         }
669
670         if (game_state_message.subject == "game state") {
671             if (game_state_message.number_of_reads > 0) {
672                 std::cout « "Game state message received by " « this « std::endl;
673                 this->message_hub.popMessage(GAME_STATE_CHANNEL);
674             }
675         }
```

```
676     }
677
678     return;
679 }   /* __processMessage() */
```

4.5.3.18 __sendCreditsEarnedMessage()

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

Helper method to format and send a credits earned message.

```
540 {
541     Message credits_earned_message;
542
543     credits_earned_message.channel = SETTLEMENT_CHANNEL;
544     credits_earned_message.subject = "credits earned";
545
546     this->message_hub.sendMessage(credits_earned_message);
547
548     std::cout « "Credits earned message sent by " « this « std::endl;
549     return;
550 }   /* __sendCreditsEarnedMessage() */
```

4.5.3.19 __sendGameStateMessage()

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

Helper method to format and send a game state message.

```
424 {
425     Message game_state_message;
426
427     game_state_message.channel = GAME_STATE_CHANNEL;
428     game_state_message.subject = "game state";
429
430     game_state_message.int_payload["year"] = this->year;
431     game_state_message.int_payload["month"] = this->month;
432     game_state_message.int_payload["population"] = this->population;
433     game_state_message.int_payload["credits"] = this->credits;
434     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
435     game_state_message.int_payload["cumulative_emissions_tonnes"] =
436         this->cumulative_emissions_tonnes;
437
438     game_state_message.int_payload["reads"] = 0;
439
440     switch (this->game_phase) {
441         case (GamePhase :: BUILD_SETTLEMENT): {
442             game_state_message.string_payload["game phase"] = "build settlement";
443
444             break;
445         }
446
447
448         case (GamePhase :: SYSTEM_MANAGEMENT): {
449             game_state_message.string_payload["game phase"] = "system management";
450
451             break;
452         }
453
454
455         case (GamePhase :: LOSS_EMISSIONS): {
456             game_state_message.string_payload["game phase"] = "loss emissions";
457
458             break;
459         }
460
461
462         case (GamePhase :: LOSS_DEMAND): {
463             game_state_message.string_payload["game phase"] = "loss demand";
```

```
464
465             break;
466         }
467
468
469         case (GamePhase :: LOSS_CREDITS): {
470             game_state_message.string_payload["game phase"] = "loss credits";
471
472             break;
473         }
474
475
476         case (GamePhase :: VICTORY): {
477             game_state_message.string_payload["game phase"] = "victory";
478
479             break;
480         }
481
482
483         default: {
484             // do nothing!
485
486             break;
487         }
488     }
489
490     game_state_message.vector_payload["demand_vec_MWh"] = this->demand_vec_MWh;
491
492     this->message_hub.sendMessage(game_state_message);
493
494     std::cout « "Game state message sent by " « this « std::endl;
495     return;
496 }   /* __sendGameStateMessage() */
```

4.5.3.20 __sendTurnAdvanceMessage()

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

Helper method to format and send a turn advance message.

```
511 {
512     Message turn_advance_message;
513
514     turn_advance_message.channel = GAME_STATE_CHANNEL;
515     turn_advance_message.subject = "turn advance";
516
517     turn_advance_message.int_payload["credits"] = this->credits;
518     turn_advance_message.int_payload["month"] = this->month;
519     turn_advance_message.int_payload["demand_MWh"] = this->demand_MWh;
520
521     this->message_hub.sendMessage(turn_advance_message);
522
523     std::cout « "Turn advance message sent by " « this « std::endl;
524     return;
525 }   /* __sendTurnAdvanceMessage() */
```

4.5.3.21 __summarizeTurn()

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

Helper method to generate end of turn summary.

```
801 {
802     if (this->turn - 1 == 0) {
803         return;
804     }
805
806     this->substring_idx = 0;
807
```

```
808     //  1. handle dispatch and demand
809     if (this->demand_served_MWh > this->past_demand_MWh) {
810         this->overproduction_MWh = this->demand_served_MWh - this->past_demand_MWh;
811         this->demand_served_MWh -= this->overproduction_MWh;
812 
813         this->overproduction_penalty =
814             round(CREDITS_PER_MWH_SERVED * this->overproduction_MWh);
815     }
816 
817     else if (this->demand_served_MWh < this->past_demand_MWh) {
818         this->demand_remaining_MWh = this->past_demand_MWh - this->demand_served_MWh;
819     }
820 
821     //  2. compute dispatch income
822     this->dispatch_income = round(CREDITS_PER_MWH_SERVED * this->demand_served_MWh);
823 
824     if (this->dispatch_income > 0) {
825         this->__sendCreditsEarnedMessage();
826     }
827 
828     //  3. compute net credit flow
829     this->net_credit_flow = this->dispatch_income -
830         this->overproduction_penalty -
831         this->turn_fuel_cost -
832         this->turn_operation_maintenance_cost;
833 
834     this->credits += this->net_credit_flow;
835 
836     //  4. assemble turn summary string
837     this->turn_summary_string.clear();
838 
839     //16 line x 32 char console  "                                \n";
840     this->turn_summary_string  = "     **** TURN ";
841     this->turn_summary_string += std::to_string(this->turn - 1);
842     this->turn_summary_string += " SUMMARY ****   \n";
843     this->turn_summary_string += "                                \n";
844 
845     this->turn_summary_string += "DEMAND:            ";
846     this->turn_summary_string += std::to_string(this->past_demand_MWh);
847     this->turn_summary_string += " MWh\n";
848 
849     this->turn_summary_string += "DEMAND SERVED:     ";
850     this->turn_summary_string += std::to_string(this->demand_served_MWh);
851     this->turn_summary_string += " MWh\n";
852 
853     if (this->overproduction_MWh > 0) {
854         this->turn_summary_string += "OVERPRODUCTION:    ";
855         this->turn_summary_string += std::to_string(this->overproduction_MWh);
856         this->turn_summary_string += " MWh\n";
857     }
858 
859     else if (this->demand_remaining_MWh > 0) {
860         this->turn_summary_string += "DEMAND REMAINING:  ";
861         this->turn_summary_string += std::to_string(this->demand_remaining_MWh);
862         this->turn_summary_string += " MWh\n";
863     }
864 
865     this->turn_summary_string += "                                \n";
866     this->turn_summary_string += "                                \n";
867 
868     this->turn_summary_string += "DISPATCH INCOME:  +";
869     this->turn_summary_string += std::to_string(this->dispatch_income);
870     this->turn_summary_string += " K\n";
871 
872     this->turn_summary_string += "FUEL COST:        -";
873     this->turn_summary_string += std::to_string(this->turn_fuel_cost);
874     this->turn_summary_string += " K\n";
875 
876     this->turn_summary_string += "OP & MAINT COST:  -";
877     this->turn_summary_string += std::to_string(this->turn_operation_maintenance_cost);
878     this->turn_summary_string += " K\n";
879 
880     this->turn_summary_string += "OVERPRODUCTION:   -";
881     this->turn_summary_string += std::to_string(this->overproduction_penalty);
882     this->turn_summary_string += " K\n";
883 
884     this->turn_summary_string += "--------------------------------\n";
885 
886     this->turn_summary_string += "NET:              ";
887 
888     if (this->net_credit_flow > 0) {
889         this->turn_summary_string += "+";
890     }
891 
892     this->turn_summary_string += std::to_string(this->net_credit_flow);
893     this->turn_summary_string += " K\n";
894 
```

```
895     this->turn_summary_string += "                                \n";
896
897     this->turn_summary_string += "EMISSIONS: ";
898     this->turn_summary_string += std::to_string(this->turn_emissions_tonnes);
899     this->turn_summary_string += " tonnes CO2e\n";
900
901     if (this->turn_emissions_tonnes <= 0) {
902         this->consecutive_zero_emissions_months++;
903     }
904
905     else {
906         this->consecutive_zero_emissions_months = 0;
907     }
908
909     //  5. send game state message
910     this->__sendGameStateMessage();
911
912     return;
913 }   /* _summarizeTurn() */
```

4.5.3.22 __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() */
```

4.5.3.23 __updatePopulation()

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

Helper method to update (i.e. grow) population.

```
143 {
144     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
145     std::default_random_engine generator(seed);
146
147     std::normal_distribution<double> normal_dist(
148         MEAN_POPULATION_GROWTH_RATE,
149         STDEV_POPULATION_GROWTH_RATE
150     );
151
152     double growth_rate = normal_dist(generator);
153
154     this->population = ceil((1 + growth_rate) * this->population);
155
156     return;
157 }   /* __updatePopulation() */
```

4.5.3.24 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).

```
1580 {
1581     //  1. play brand animation
1582     //...
1583
1584     //  2. show splash screen
1585     //...
1586
1587     //  3. start game loop
1588     while (not this->game_loop_broken) {
1589         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
1590
1591         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
1592             //  6.1. process events
1593             while (this->render_window_ptr->pollEvent(this->event)) {
1594                 if (
1595                     (this->game_phase == GamePhase :: BUILD_SETTLEMENT) or
1596                     (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT)
1597                 ) {
1598                     this->hex_map_ptr->processEvent();
1599                     this->context_menu_ptr->processEvent();
1600                     this->__processEvent();
1601                 }
1602
1603                 else {
1604                     if (this->event.type == sf::Event::KeyPressed) {
1605                         this->game_loop_broken = true;
1606                     }
1607                 }
1608             }
1609
1610
1611             //  6.2. process messages
1612             while (this->message_hub.hasTraffic()) {
1613                 this->hex_map_ptr->processMessage();
1614                 this->context_menu_ptr->processMessage();
1615                 this->__processMessage();
1616
1617                 this->check_terminating_conditions = true;
1618
1619                 if (not this->message_deadlock) {
1620                     this->message_deadlock_frame++;
1621
1622                     if (this->message_deadlock_frame > 5 * FRAMES_PER_SECOND) {
1623                         this->message_hub.printState();
1624                         this->message_deadlock = true;
1625                     }
1626                 }
1627             }
1628             this->message_deadlock = false;
1629             this->message_deadlock_frame = 0;
1630
1631
1632             //  6.3. handle turn end summary
1633             if (this->turn_end) {
1634                 std::cout « "**** END OF TURN " « std::to_string(this->turn - 1) «
1635                     " ****" « std::endl;
1636
1637                 this->__summarizeTurn();
1638
1639                 this->turn_end = false;
1640             }
1641
1642
1643             //  6.4. check terminating conditions
1644             if (this->check_terminating_conditions) {
1645                 this->__checkTerminatingConditions();
1646                 this->check_terminating_conditions = false;
1647             }
1648
1649
1650             //  6.5. draw frame
```

```
1651             this->render_window_ptr->clear();
1652
1653             this->hex_map_ptr->draw();
1654             this->context_menu_ptr->draw();
1655             this->__draw();
1656
1657             this->render_window_ptr->display();
1658
1659
1660             //  6.6. increment frame
1661             this->frame++;
1662         }
1663
1664         // check track status, move to next if stopped
1665         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
1666             this->assets_manager_ptr->nextTrack();
1667             this->assets_manager_ptr->playTrack();
1668         }
1669
1670     }
1671
1672     return this->quit_game;
1673 }   /* 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 consecutive_zero_emissions_months

```
int Game::consecutive_zero_emissions_months
```

The number of recent, consecutive zero emission months.

4.5.4.5 context_menu_ptr

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

Pointer to the context menu.

4.5.4.6 credits

```
int Game::credits
```

Current balance of credits.

4.5.4.7 cumulative_emissions_tonnes

```
int Game::cumulative_emissions_tonnes
```

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

4.5.4.8 demand_MWh

```
int Game::demand_MWh
```

Current energy demand [MWh].

4.5.4.9 demand_remaining_MWh

```
int Game::demand_remaining_MWh
```

The demand remaining at the end of a turn.

4.5.4.10 demand_served_MWh

```
int Game::demand_served_MWh
```

The demand served at the end of a turn.

4.5.4.11 demand_vec_MWh

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

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

4.5.4.12 dispatch_income

```
int Game::dispatch_income
```

The amount earned from dispatch at the end of a turn.

4.5.4.13 event

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

The game events class.

4.5.4.14 frame

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

The current frame of the game.

4.5.4.15 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.16 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.17 hex_map_ptr

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

Pointer to the hex map (defines game world).

4.5.4.18 message_deadlock

```
bool Game::message_deadlock
```

A boolean indicating whether a message deadlock has been detected.

4.5.4.19 message_deadlock_frame

```
int Game::message_deadlock_frame
```

A frame counter for detecting message deadlock.

4.5.4.20 message_hub

```
MessageHub Game::message_hub
```

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

4.5.4.21 month

```
int Game::month
```

Current game month.

4.5.4.22 net_credit_flow

```
int Game::net_credit_flow
```

The net credit flow at the end of a turn.

4.5.4.23 overproduction_MWh

```
int Game::overproduction_MWh
```

The amount of overproduction at the end of a turn.

4.5.4.24 overproduction_penalty

```
int Game::overproduction_penalty
```

The penalty for overproduction.

4.5.4.25 past_demand_MWh

```
int Game::past_demand_MWh
```

The demand in the previous turn.

4.5.4.26 population

```
int Game::population
```

Current population.

4.5.4.27 quit_game

```
bool Game::quit_game
```

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

4.5.4.28 render_window_ptr

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

A pointer to the render window.

4.5.4.29 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

4.5.4.30 show_tutorial

```
bool Game::show_tutorial
```

A boolean indicating whether or not to show the tutorial.

4.5.4.31 substring_idx

```
size_t Game::substring_idx
```

The index of the turn summary substring.

4.5.4.32 time_since_start_s

```
double Game::time_since_start_s
```

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

4.5.4.33 turn

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

The current game turn.

4.5.4.34 turn_emissions_tonnes

```
int Game::turn_emissions_tonnes
```

The amount of emissions at the end of a turn.

4.5.4.35 turn_end

```
bool Game::turn_end
```

A boolean indicating a turn end.

4.5.4.36 turn_fuel_cost

```
int Game::turn_fuel_cost
```

The cost of fuel at the end of a turn.

4.5.4.37 turn_operation_maintenance_cost

```
int Game::turn_operation_maintenance_cost
```

The cost of operation and maintenance at the end of a turn.

4.5.4.38 turn_summary_string

```
std::string Game::turn_summary_string
```

A string representation of the end of turn summary.

4.5.4.39 turn_summary_text

```
sf::Text Game::turn_summary_text
```

A text representation (drawable) of the end of turn summary.

4.5.4.40 year

```
int Game::year
```

Current game year.

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

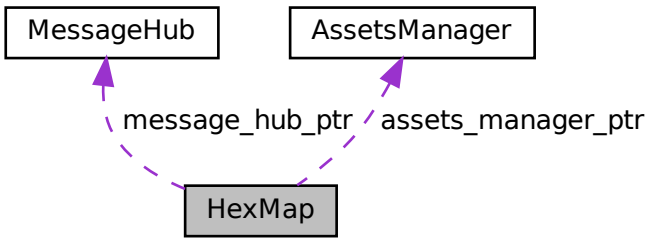
- header/Game.h
- source/Game.cpp

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.
- bool settlement_position_logged

  A boolean which indicates if the settlement position has been logged.
- 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.
- int demand_MWh

  Current energy demand [MWh].
- 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.
- double settlement_position_x

  The x position of the settlement.
- double settlement_position_y

  The y position of the settlement.
- 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 __logSettlementPosition (void)

  Helper method to log settlement position (if not already done).
- 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.
- void __drawTotalDispatch (void)

  Helper method to compute and draw current total production / dispatch from all production assets.

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.

```
1317 {
1318     //  1. set attributes
1319
1320     //  1.1. private
1321     this->event_ptr = event_ptr;
1322     this->render_window_ptr = render_window_ptr;
1323
1324     this->assets_manager_ptr = assets_manager_ptr;
1325     this->message_hub_ptr = message_hub_ptr;
1326
1327     //  1.2. public
1328     this->show_resource = false;
1329     this->tile_selected = false;
1330     this->settlement_position_logged = false;
1331
1332     this->frame = 0;
1333
1334     this->n_layers = n_layers;
1335     if (this->n_layers < 0) {
1336         this->n_layers = 0;
1337     }
1338
1339     this->demand_MWh = 0;
1340
1341     this->position_x = 400;
1342     this->position_y = 400;
1343
1344     this->settlement_position_x = 0;
1345     this->settlement_position_y = 0;
1346
1347     //  2. assemble n layer hex map
1348     this->__assembleHexMap();
1349
1350     //  3. set up and position drawable attributes
1351     this->__setUpGlassScreen();
1352
1353     //  4. add message channel(s)
1354     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1355     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1356     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1357     this->message_hub_ptr->addChannel(HEX_MAP_CHANNEL);
1358
1359     std::cout « "HexMap constructed at " « this « std::endl;
1360
1361     return;
1362 }   /* HexMap(), intended */
```

4.6.2.2 ~HexMap()

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

Destructor for the HexMap class.

```
1689 {
1690     this->clear();
1691
1692     std::cout « "HexMap at " « this « " destroyed" « std::endl;
1693
1694     return;
1695 }   /* ~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.

```
1123 {
1124     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
1125
```

```
1126     for (size_t i = 0; i < neighbours_vec.size(); i++) {
1127         neighbours_vec[i]->assess();
1128     }
1129
1130     return;
1131 }   /* __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 __drawTotalDispatch()

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

Helper method to compute and draw current total production / dispatch from all production assets.

```
1147 {
1148     //  1. compute total production / dispatch
1149     int total_production_dispatch_MWh = 0;
1150 
1151     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1152     std::map<double, HexTile*>::iterator hex_map_iter_y;
1153 
1154     TileImprovement* tile_improvement_ptr;
1155 
1156     for (
1157         hex_map_iter_x = this->hex_map.begin();
1158         hex_map_iter_x != this->hex_map.end();
1159         hex_map_iter_x++
1160     ) {
1161         for (
1162             hex_map_iter_y = hex_map_iter_x->second.begin();
1163             hex_map_iter_y != hex_map_iter_x->second.end();
1164             hex_map_iter_y++
1165         ) {
1166             if (
1167                 (hex_map_iter_y->second->has_improvement) and
1168                 (hex_map_iter_y->second->tile_improvement_ptr->tile_improvement_type !=
1169                     TileImprovementType :: SETTLEMENT)
1170             ) {
1171                 tile_improvement_ptr = hex_map_iter_y->second->tile_improvement_ptr;
1172 
1173                 switch (tile_improvement_ptr->tile_improvement_type) {
1174                     case (TileImprovementType :: DIESEL_GENERATOR): {
1175                         total_production_dispatch_MWh +=
1176                             ((DieselGenerator*)tile_improvement_ptr)->production_MWh;
1177 
1178                         break;
1179                     }
1180 
1181 
1182                     case (TileImprovementType :: SOLAR_PV): {
1183                         total_production_dispatch_MWh +=
1184                             ((SolarPV*)tile_improvement_ptr)->dispatch_MWh;
1185 
1186                         break;
1187                     }
1188 
1189 
1190                     case (TileImprovementType :: TIDAL_TURBINE): {
1191                         total_production_dispatch_MWh +=
1192                             ((TidalTurbine*)tile_improvement_ptr)->dispatch_MWh;
1193 
1194                         break;
1195                     }
1196 
1197 
1198                     case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
1199                         total_production_dispatch_MWh +=
1200                             ((WaveEnergyConverter*)tile_improvement_ptr)->dispatch_MWh;
1201 
1202                         break;
1203                     }
1204 
1205 
1206                     case (TileImprovementType :: WIND_TURBINE): {
1207                         total_production_dispatch_MWh +=
1208                             ((WindTurbine*)tile_improvement_ptr)->dispatch_MWh;
1209 
1210                         break;
1211                     }
1212 
1213 
1214                     default: {
1215                         // do nothing!
1216 
1217                         break;
1218                     }
1219                 }
1220             }
1221         }
1222     }
1223 
1224     //  2. construct total text
```

```
1225     sf::Text total_production_dispatch_text(
1226         std::to_string(total_production_dispatch_MWh),
1227         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1228         16
1229     );
1230 
1231     total_production_dispatch_text.setOrigin(
1232         total_production_dispatch_text.getLocalBounds().width / 2,
1233         total_production_dispatch_text.getLocalBounds().height / 2
1234     );
1235 
1236     total_production_dispatch_text.setPosition(800 - 20, 20 - 4);
1237 
1238     sf::Color text_colour;
1239 
1240     if (total_production_dispatch_MWh < this->demand_MWh) {
1241         text_colour = MONOCHROME_TEXT_RED;
1242     }
1243 
1244     else if (total_production_dispatch_MWh > this->demand_MWh) {
1245         text_colour = MONOCHROME_TEXT_AMBER;
1246     }
1247 
1248     else {
1249         text_colour = MONOCHROME_TEXT_GREEN;
1250     }
1251 
1252     total_production_dispatch_text.setFillColor(text_colour);
1253 
1254     //  4. construct total backing
1255     sf::RectangleShape total_production_dispatch_backing(sf::Vector2f(32, 32));
1256 
1257     total_production_dispatch_backing.setOrigin(
1258         total_production_dispatch_backing.getLocalBounds().width / 2,
1259         total_production_dispatch_backing.getLocalBounds().height / 2
1260     );
1261 
1262     total_production_dispatch_backing.setPosition(800 - 20, 20);
1263 
1264     total_production_dispatch_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
1265 
1266     total_production_dispatch_backing.setOutlineColor(MENU_FRAME_GREY);
1267     total_production_dispatch_backing.setOutlineThickness(2);
1268 
1269     //  4. draw
1270     if (total_production_dispatch_MWh > 0) {
1271         this->render_window_ptr->draw(total_production_dispatch_backing);
1272         this->render_window_ptr->draw(total_production_dispatch_text);
1273     }
1274 
1275     return;
1276 }   /* __drawTotalDispatch() */
```

4.6.3.5 __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() */
```

4.6.3.6 __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() */
```

4.6.3.7 __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() */
```

4.6.3.8 __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() */
```

4.6.3.9 __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() */
```

4.6.3.10 __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() */
```

4.6.3.11 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
1015 {
1016     switch (this->event_ptr->key.code) {
1017         case (sf::Keyboard::Escape): {
1018             this->tile_selected = false;
1019         }
```

```
1020
1021
1022         default: {
1023             // do nothing!
1024
1025             break;
1026         }
1027     }
1028
1029     return;
1030 }   /* __handleKeyPressEvents() */
```

4.6.3.12 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
1045 {
1046     switch (this->event_ptr->mouseButton.button) {
1047         case (sf::Mouse::Left): {
1048             HexTile* hex_ptr = this->__getSelectedTile();
1049
1050             if (hex_ptr != NULL) {
1051                 this->tile_selected = true;
1052             }
1053
1054             else if (this->tile_selected) {
1055                 this->tile_selected = false;
1056                 this->__sendNoTileSelectedMessage();
1057             }
1058
1059             break;
1060         }
1061
1062
1063         case (sf::Mouse::Right): {
1064             if (this->tile_selected) {
1065                 this->tile_selected = false;
1066                 this->__sendNoTileSelectedMessage();
1067             }
1068
1069             break;
1070         }
1071
1072
1073         default: {
1074             // do nothing!
1075
1076             break;
1077         }
1078     }
1079
1080     return;
1081 }   /* __handleMouseButtonEvents() */
```

4.6.3.13 __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() */
```

4.6.3.14 __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() */
```

4.6.3.15 __logSettlementPosition()

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

Helper method to log settlement position (if not already done).

```
959 {
960     bool break_flag = false;
961 
962     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
963     std::map<double, HexTile*>::iterator hex_map_iter_y;
964 
965     for (
966         hex_map_iter_x = this->hex_map.begin();
967         hex_map_iter_x != this->hex_map.end();
968         hex_map_iter_x++
969     ) {
970         for (
971             hex_map_iter_y = hex_map_iter_x->second.begin();
972             hex_map_iter_y != hex_map_iter_x->second.end();
973             hex_map_iter_y++
974         ) {
975             if (
976                 (hex_map_iter_y->second->has_improvement) and
977                 (hex_map_iter_y->second->tile_improvement_ptr->tile_improvement_type ==
978                     TileImprovementType :: SETTLEMENT)
979             ) {
980                 this->settlement_position_x = hex_map_iter_y->second->position_x;
981                 this->settlement_position_y = hex_map_iter_y->second->position_y;
982 
983                 this->settlement_position_logged = true;
984 
985                 std::cout « "Settlement position logged, (" «
986                     this->settlement_position_x « ", " «
987                     this->settlement_position_y « ")" « std::endl;
988 
989                 break_flag = true;
990                 break;
991             }
992         }
993 
994         if (break_flag) {
995             break;
996         }
997     }
998 
999     return;
1000 }   /* __logSettlementPosition() */
```

4.6.3.16 __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() */
```

4.6.3.17 __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() */
```

4.6.3.18 __sendNoTileSelectedMessage()

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

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

```
1096 {
1097     Message no_tile_selected_message;
1098
1099     no_tile_selected_message.channel = NO_TILE_SELECTED_CHANNEL;
1100     no_tile_selected_message.subject = "no tile selected";
1101
1102     this->message_hub_ptr->sendMessage(no_tile_selected_message);
1103
1104     std::cout « "No tile selected message sent by " « this « std::endl;
1105     return;
1106 }   /* __sendNoTileSelectedMessage() */
```

4.6.3.19 __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() */
```

4.6.3.20 __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() */
```

4.6.3.21 assess()

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

Method to assess the resource of the selected tile.

```
1377 {
1378     HexTile* selected_tile_ptr = this->__getSelectedTile();
1379     if (selected_tile_ptr != NULL) {
1380         selected_tile_ptr->assess();
1381     }
1382
1383     return;
1384 }   /* assess() */
```

4.6.3.22 clear()

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

Method to clear the hex map.

```
1651 {
1652     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1653     std::map<double, HexTile*>::iterator hex_map_iter_y;
1654     for (
1655         hex_map_iter_x = this->hex_map.begin();
1656         hex_map_iter_x != this->hex_map.end();
1657         hex_map_iter_x++
1658     ) {
1659         for (
```

```
1660             hex_map_iter_y = hex_map_iter_x->second.begin();
1661             hex_map_iter_y != hex_map_iter_x->second.end();
1662             hex_map_iter_y++
1663         ) {
1664             delete hex_map_iter_y->second;
1665         }
1666     }
1667     this->hex_map.clear();
1668
1669     this->tile_position_x_vec.clear();
1670     this->tile_position_y_vec.clear();
1671     this->border_tiles_vec.clear();
1672
1673     return;
1674 }   /* clear() */
```

4.6.3.23 draw()

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

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

```
1575 {
1576     //  1. draw background
1577     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1578     glass_screen_colour.a = 255;
1579     this->glass_screen.setFillColor(glass_screen_colour);
1580
1581     this->render_window_ptr->draw(this->glass_screen);
1582
1583     //  2. draw tiles (other than the selected tile) in drawing order
1584     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1585         if (not this->hex_draw_order_vec[i]->is_selected) {
1586             this->hex_draw_order_vec[i]->draw();
1587         }
1588     }
1589
1590     //  3. draw total production / dispatch overlay
1591     if (this->settlement_position_logged) {
1592         this->__drawTotalDispatch();
1593     }
1594
1595     //  4. draw selected tile
1596     HexTile* selected_tile_ptr = this->__getSelectedTile();
1597     if (selected_tile_ptr != NULL) {
1598         selected_tile_ptr->draw();
1599
1600         if (
1601             (selected_tile_ptr->has_improvement) and
1602             (selected_tile_ptr->tile_improvement_ptr->tile_improvement_type ==
1603                 TileImprovementType :: SETTLEMENT)
1604         ) {
1605             this->__drawTotalDispatch();
1606         }
1607     }
1608
1609     //  5. draw resource overlay text indication
1610     if (this->show_resource) {
1611         sf::Text resource_overlay_text(
1612             "**** RENEWABLE RESOURCE OVERLAY ****",
1613             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1614             16
1615         );
1616
1617         resource_overlay_text.setPosition(
1618             (800 - resource_overlay_text.getLocalBounds().width) / 2,
1619             GAME_HEIGHT - 70
1620         );
1621
1622         resource_overlay_text.setFillColor(MONOCHROME_TEXT_GREEN);
1623
1624         this->render_window_ptr->draw(resource_overlay_text);
1625     }
1626
1627     //  6. draw glass screen
1628     glass_screen_colour = this->glass_screen.getFillColor();
1629     glass_screen_colour.a = 40;
1630     this->glass_screen.setFillColor(glass_screen_colour);
```

```
1631
1632     this->render_window_ptr->draw(this->glass_screen);
1633
1634     this->frame++;
1635     return;
1636 }    /* draw() */
```

4.6.3.24 processEvent()

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

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

```
1462 {
1463     //  1. process HexTile events
1464     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1465     std::map<double, HexTile*>::iterator hex_map_iter_y;
1466     for (
1467         hex_map_iter_x = this->hex_map.begin();
1468         hex_map_iter_x != this->hex_map.end();
1469         hex_map_iter_x++
1470     ) {
1471         for (
1472             hex_map_iter_y = hex_map_iter_x->second.begin();
1473             hex_map_iter_y != hex_map_iter_x->second.end();
1474             hex_map_iter_y++
1475         ) {
1476             hex_map_iter_y->second->processEvent();
1477         }
1478     }
1479
1480     //  2. process HexMap events
1481     if (this->event_ptr->type == sf::Event::KeyPressed) {
1482         this->__handleKeyPressEvents();
1483     }
1484
1485     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1486         this->__handleMouseButtonEvents();
1487     }
1488
1489     return;
1490 }    /* processEvent() */
```

4.6.3.25 processMessage()

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

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

```
1505 {
1506     //  1. process HexTile messages
1507     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1508     std::map<double, HexTile*>::iterator hex_map_iter_y;
1509     for (
1510         hex_map_iter_x = this->hex_map.begin();
1511         hex_map_iter_x != this->hex_map.end();
1512         hex_map_iter_x++
1513     ) {
1514         for (
1515             hex_map_iter_y = hex_map_iter_x->second.begin();
1516             hex_map_iter_y != hex_map_iter_x->second.end();
1517             hex_map_iter_y++
1518         ) {
1519             hex_map_iter_y->second->processMessage();
1520         }
1521     }
1522
1523     //  2. process HexMap messages
1524     if (not this->message_hub_ptr->isEmpty(HEX_MAP_CHANNEL)) {
```

```
1525         Message hex_map_message = this->message_hub_ptr->receiveMessage(
1526             HEX_MAP_CHANNEL
1527         );
1528 
1529         if (hex_map_message.subject == "assess neighbours") {
1530             HexTile* hex_ptr = this->__getSelectedTile();
1531             this->__assessNeighbours(hex_ptr);
1532 
1533             std::cout << "Assess neighbours message received by " << this << std::endl;
1534             this->message_hub_ptr->popMessage(HEX_MAP_CHANNEL);
1535         }
1536     }
1537 
1538     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
1539         Message game_state_message = this->message_hub_ptr->receiveMessage(
1540             GAME_STATE_CHANNEL
1541         );
1542 
1543         if (game_state_message.subject == "game state") {
1544             this->demand_MWh = game_state_message.int_payload["demand_MWh"];
1545 
1546             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
1547 
1548             std::cout << "Game state message read and passed by " << this <<
1549                 " (demand: " << this->demand_MWh << " MWh)" << std::endl;
1550         }
1551     }
1552 
1553     //  3. log settlement position (if applicable)
1554     if (not this->settlement_position_logged) {
1555         this->__logSettlementPosition();
1556     }
1557 
1558     return;
1559 }   /* processMessage() */
```

4.6.3.26 reroll()

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

Method to re-roll the hex map.

```
1399 {
1400     this->clear();
1401     this->__assembleHexMap();
1402 
1403     return;
1404 }   /* reroll() */
```

4.6.3.27 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
1419 {
1420     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1421     std::map<double, HexTile*>::iterator hex_map_iter_y;
1422     for (
1423         hex_map_iter_x = this->hex_map.begin();
1424         hex_map_iter_x != this->hex_map.end();
1425         hex_map_iter_x++
1426     ) {
1427         for (
1428             hex_map_iter_y = hex_map_iter_x->second.begin();
1429             hex_map_iter_y != hex_map_iter_x->second.end();
1430             hex_map_iter_y++
1431         ) {
1432             hex_map_iter_y->second->toggleResourceOverlay();
```

```
1433         }
1434     }
1435
1436     if (this->show_resource) {
1437         this->show_resource = false;
1438         this->assets_manager_ptr->getSound("resource overlay toggle off")->play();
1439     }
1440
1441     else {
1442         this->show_resource = true;
1443         this->assets_manager_ptr->getSound("resource overlay toggle on")->play();
1444     }
1445
1446     return;
1447 }   /* 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 demand_MWh

```
int HexMap::demand_MWh
```

Current energy demand [MWh].

4.6.4.4 event_ptr

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

A pointer to the event class.

4.6.4.5 frame

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

The current frame of this object.

4.6.4.6 glass_screen

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

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

4.6.4.7 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

4.6.4.8 hex_map

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

A position-indexed, nested map of hex tiles.

4.6.4.9 message_hub_ptr

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

A pointer to the message hub.

4.6.4.10 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

4.6.4.11 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

4.6.4.12 position_x

```
double HexMap::position_x
```

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

4.6.4.13 position_y

```
double HexMap::position_y
```

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

4.6.4.14 render_window_ptr

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

A pointer to the render window.

4.6.4.15 settlement_position_logged

```
bool HexMap::settlement_position_logged
```

A boolean which indicates if the settlement position has been logged.

4.6.4.16 settlement_position_x

```
double HexMap::settlement_position_x
```

The x position of the settlement.

4.6.4.17 settlement_position_y

```
double HexMap::settlement_position_y
```

The y position of the settlement.

4.6.4.18 show_resource

```
bool HexMap::show_resource
```

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

4.6.4.19 tile_position_x_vec

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

A vector of tile x positions.

4.6.4.20 tile_position_y_vec

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

A vector of tile y position.

4.6.4.21 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:

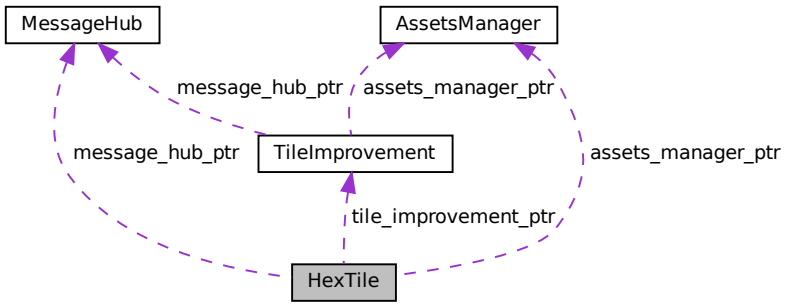
- header/HexMap.h
- source/HexMap.cpp

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.

```
2308 {
2309     //  1. set attributes
2310
2311     //  1.1. private
2312     this->event_ptr = event_ptr;
2313     this->render_window_ptr = render_window_ptr;
2314
2315     this->assets_manager_ptr = assets_manager_ptr;
2316     this->message_hub_ptr = message_hub_ptr;
2317
2318     //  1.2. public
2319     this->show_node = false;
2320     this->show_resource = false;
2321     this->resource_assessed = false;
2322     this->resource_assessment = false;
2323     this->is_selected = false;
2324     this->draw_explosion = false;
2325
2326     this->decoration_cleared = false;
2327     this->has_improvement = false;
2328     this->tile_improvement_ptr = NULL;
2329
2330     this->build_menu_open = false;
2331
2332     this->explosion_frame = 0;
2333
2334     this->frame = 0;
2335     this->credits = 0;
2336
2337     this->scrap_improvement_frame = 0;
2338
2339     this->position_x = position_x;
2340     this->position_y = position_y;
2341
2342     this->major_radius = 32;
2343     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2344
2345     this->game_phase = "build settlement";
2346
2347     //  2. set up and position drawable attributes
2348     this->__setUpNodeSprite();
2349     this->__setUpTileSprite();
2350     this->__setUpSelectOutlineSprite();
2351     this->__setUpResourceChipSprite();
2352     this->__setResourceText();
2353     this->__setUpMagnifyingGlassSprite();
2354     this->__setUpTileExplosionReel();
2355
2356     //  3. set tile type and resource (default to none type and average)
2357     this->setTileType(TileType :: NONE_TYPE);
2358     this->setTileResource(TileResource :: AVERAGE);
2359
2360     std::cout « "HexTile constructed at " « this « std::endl;
2361
2362     return;
2363 }   /* HexTile() */
```

4.7.2.2 ~HexTile()

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

Destructor for the HexTile class.

```
2931 {
2932     if (this->tile_improvement_ptr != NULL) {
2933         delete this->tile_improvement_ptr;
2934     }
2935
2936     std::cout « "HexTile at " « this « " destroyed" « std::endl;
2937
2938     return;
2939 }   /* ~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.

```
1409 {
1410     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1411
1412     if (this->credits < build_cost) {
1413         std::cout « "Cannot build diesel generator: insufficient credits (need "
1414             « build_cost « " K)" « std::endl;
1415
1416         this->__sendInsufficientCreditsMessage();
1417         return;
1418     }
1419
1420     this->tile_improvement_ptr = new DieselGenerator(
1421         this->position_x,
1422         this->position_y,
1423         this->tile_resource,
1424         this->event_ptr,
1425         this->render_window_ptr,
1426         this->assets_manager_ptr,
1427         this->message_hub_ptr
1428     );
1429
1430     this->has_improvement = true;
1431     this->__closeBuildMenu();
1432
1433     this->__sendCreditsSpentMessage(build_cost);
1434     this->__sendTileStateMessage();
1435     this->__sendGameStateRequest();
1436
1437     return;
1438 }   /* __buildDieselGenerator() */
```

4.7.3.2 __buildEnergyStorage()

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

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

```
1657 {
1658     /*
1659     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1660
1661     if (this->credits < build_cost) {
1662         std::cout « "Cannot build energy storage system: insufficient credits (need "
1663             « build_cost « " K)" « std::endl;
1664
1665         this->__sendInsufficientCreditsMessage();
1666         return;
1667     }
1668
1669     this->tile_improvement_ptr = new EnergyStorageSystem(
1670         this->position_x,
1671         this->position_y,
1672         this->event_ptr,
1673         this->render_window_ptr,
1674         this->assets_manager_ptr,
1675         this->message_hub_ptr
1676     );
1677
1678     this->has_improvement = true;
1679     this->__closeBuildMenu();
1680
1681     this->__sendCreditsSpentMessage(build_cost);
1682     this->__sendTileStateMessage();
1683     this->__sendGameStateRequest();
1684     */
1685     return;
1686 }   /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

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

Helper method to build a settlement on this tile.

```
1362 {
1363     if (this->credits < BUILD_SETTLEMENT_COST) {
1364         std::cout « "Cannot build settlement: insufficient credits (need "
1365             « BUILD_SETTLEMENT_COST « " K)" « std::endl;
1366 
1367         this->__sendInsufficientCreditsMessage();
1368         return;
1369     }
1370 
1371     this->__clearDecoration();
1372 
1373     this->tile_improvement_ptr = new Settlement(
1374         this->position_x,
1375         this->position_y,
1376         this->tile_resource,
1377         this->event_ptr,
1378         this->render_window_ptr,
1379         this->assets_manager_ptr,
1380         this->message_hub_ptr
1381     );
1382 
1383     this->has_improvement = true;
1384 
1385     this->assess();
1386     this->__sendAssessNeighboursMessage();
1387 
1388     this->__sendUpdateGamePhaseMessage("system management");
1389     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1390     this->__sendTileStateMessage();
1391     this->__sendGameStateRequest();
1392 
1393     return;
1394 }   /* __buildSettlement() */
```

4.7.3.4 __buildSolarPV()

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

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

```
1453 {
1454     int build_cost = SOLAR_PV_BUILD_COST;
1455 
1456     if (this->tile_type == TileType :: LAKE) {
1457         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1458     }
1459 
1460     if (this->credits < build_cost) {
1461         std::cout « "Cannot build solar PV array: insufficient credits (need "
1462             « build_cost « " K)" « std::endl;
1463 
1464         this->__sendInsufficientCreditsMessage();
1465         return;
1466     }
1467 
1468     this->tile_improvement_ptr = new SolarPV(
1469         this->position_x,
1470         this->position_y,
1471         this->tile_resource,
1472         this->event_ptr,
1473         this->render_window_ptr,
1474         this->assets_manager_ptr,
1475         this->message_hub_ptr
1476     );
1477 
1478     this->has_improvement = true;
1479     this->__closeBuildMenu();
1480 
1481     if (this->tile_type == TileType :: LAKE) {
```

```
1482         this->decoration_cleared = true;
1483         this->assets_manager_ptr->getSound("splash")->play();
1484     }
1485 
1486     this->__sendCreditsSpentMessage(build_cost);
1487     this->__sendTileStateMessage();
1488     this->__sendGameStateRequest();
1489 
1490     return;
1491 }   /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

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

Helper method to build a tidal turbine on this tile.

```
1565 {
1566     int build_cost = TIDAL_TURBINE_BUILD_COST;
1567 
1568     if (this->credits < build_cost) {
1569         std::cout << "Cannot build tidal turbine: insufficient credits (need "
1570             << build_cost << " K)" << std::endl;
1571 
1572         this->__sendInsufficientCreditsMessage();
1573         return;
1574     }
1575 
1576     this->tile_improvement_ptr = new TidalTurbine(
1577         this->position_x,
1578         this->position_y,
1579         this->tile_resource,
1580         this->event_ptr,
1581         this->render_window_ptr,
1582         this->assets_manager_ptr,
1583         this->message_hub_ptr
1584     );
1585 
1586     this->has_improvement = true;
1587     this->decoration_cleared = true;
1588     this->assets_manager_ptr->getSound("splash")->play();
1589     this->__closeBuildMenu();
1590 
1591     this->__sendCreditsSpentMessage(build_cost);
1592     this->__sendTileStateMessage();
1593     this->__sendGameStateRequest();
1594 
1595     return;
1596 }   /* __buildTidalTurbine() */
```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1611 {
1612     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1613 
1614     if (this->credits < build_cost) {
1615         std::cout << "Cannot build wave energy converter: insufficient credits (need "
1616             << build_cost << " K)" << std::endl;
1617 
1618         this->__sendInsufficientCreditsMessage();
1619         return;
1620     }
1621 
1622     this->tile_improvement_ptr = new WaveEnergyConverter(
1623         this->position_x,
```

```
1624         this->position_y,
1625         this->tile_resource,
1626         this->event_ptr,
1627         this->render_window_ptr,
1628         this->assets_manager_ptr,
1629         this->message_hub_ptr
1630     );
1631
1632     this->has_improvement = true;
1633     this->decoration_cleared = true;
1634     this->assets_manager_ptr->getSound("splash")->play();
1635     this->__closeBuildMenu();
1636
1637     this->__sendCreditsSpentMessage(build_cost);
1638     this->__sendTileStateMessage();
1639     this->__sendGameStateRequest();
1640
1641     return;
1642 }   /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1506 {
1507     int build_cost = WIND_TURBINE_BUILD_COST;
1508
1509     if (
1510         (this->tile_type == TileType :: LAKE) or
1511         (this->tile_type == TileType :: OCEAN)
1512     ) {
1513         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1514     }
1515
1516     if (this->credits < build_cost) {
1517         std::cout « "Cannot build wind turbine: insufficient credits (need "
1518             « build_cost « " K)" « std::endl;
1519
1520         this->__sendInsufficientCreditsMessage();
1521         return;
1522     }
1523
1524     this->tile_improvement_ptr = new WindTurbine(
1525         this->position_x,
1526         this->position_y,
1527         this->tile_resource,
1528         this->event_ptr,
1529         this->render_window_ptr,
1530         this->assets_manager_ptr,
1531         this->message_hub_ptr
1532     );
1533
1534     this->has_improvement = true;
1535     this->__closeBuildMenu();
1536
1537     if (
1538         (this->tile_type == TileType :: LAKE) or
1539         (this->tile_type == TileType :: OCEAN)
1540     ) {
1541         this->decoration_cleared = true;
1542         this->assets_manager_ptr->getSound("splash")->play();
1543     }
1544
1545     this->__sendCreditsSpentMessage(build_cost);
1546     this->__sendTileStateMessage();
1547     this->__sendGameStateRequest();
1548
1549     return;
1550 }   /* __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.

```
1337 {
1338     if (not this->build_menu_open) {
1339         return;
1340     }
1341 
1342     this->build_menu_open = false;
1343     this->assets_manager_ptr->getSound("build menu close")->play();
1344 
1345     return;
1346 }   /* __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.

```
1803 {
1804     std::string coords_substring = "TILE COORDS:  (";
1805     coords_substring += std::to_string(int(this->position_x - 400));
1806     coords_substring += ", ";
1807     coords_substring += std::to_string(int(this->position_y - 400));
1808     coords_substring += ")\n";
1809 
1810     return coords_substring;
1811 }   /* __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.

```
1962 {
1963     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1964 
1965     if (this->has_improvement) {
1966         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1967         improvement_substring += "\n";
1968     }
1969 
1970     else {
1971         improvement_substring += "NONE\n";
1972     }
1973 
1974     return improvement_substring;
1975 }   /* __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.

```
1992 {
1993     //                  32 char x 17 line console "--------------------------------\n";
1994     std::string options_substring               = "     **** TILE OPTIONS ****     \n";
1995     options_substring                          += "                                \n";
1996 
1997     if (this->game_phase == "build settlement") {
1998         if (
1999             (this->tile_type != TileType :: OCEAN) and
2000             (this->tile_type != TileType :: LAKE)
2001         ) {
2002             options_substring += "[B]:  BUILD SETTLEMENT (";
2003             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
2004             options_substring += " K)\n";
2005         }
```

```
2006     }
2007
2008
2009     else if (this->game_phase == "system management") {
2010         if (this->has_improvement) {
2011             options_substring.clear();
2012             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2013         }
2014
2015
2016         else if (not this->resource_assessed) {
2017             options_substring += "[A]:  ASSESS RESOURCE (";
2018             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2019             options_substring += " K)\n";
2020         }
2021
2022
2023         else if (
2024             (not this->decoration_cleared) and
2025             (this->tile_type != TileType :: OCEAN) and
2026             (this->tile_type != TileType :: LAKE)
2027         ) {
2028             options_substring += "[C]:  CLEAR TILE (";
2029
2030             switch (this->tile_type) {
2031                 case (TileType :: FOREST): {
2032                     options_substring += std::to_string(CLEAR_FOREST_COST);
2033
2034                     break;
2035                 }
2036
2037
2038                 case (TileType :: MOUNTAINS): {
2039                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2040
2041                     break;
2042                 }
2043
2044
2045                 case (TileType :: PLAINS): {
2046                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2047
2048                     break;
2049                 }
2050
2051
2052                 default: {
2053                     //do nothing!
2054
2055                     break;
2056                 }
2057             }
2058
2059             options_substring += " K)\n";
2060         }
2061
2062
2063         else if (
2064             (this->decoration_cleared) or
2065             (this->tile_type == TileType :: OCEAN) or
2066             (this->tile_type == TileType :: LAKE)
2067         ) {
2068             options_substring += "[B]: OPEN BUILD MENU\n";
2069         }
2070     }
2071
2072
2073     else if (this->game_phase == "victory") {
2074         options_substring                       += "       **** VICTORY ****        \n";
2075     }
2076
2077
2078     else {
2079         options_substring                       += "        **** LOSS ****          \n";
2080     }
2081
2082     return options_substring;
2083 }    /* __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.

```
1892 {
1893     std::string resource_substring = "TILE RESOURCE:     ";
1894 
1895     if (this->resource_assessed) {
1896         switch (this->tile_resource) {
1897             case (TileResource :: POOR): {
1898                 resource_substring += "POOR\n";
1899 
1900                 break;
1901             }
1902 
1903 
1904             case (TileResource ::BELOW_AVERAGE): {
1905                 resource_substring += "BELOW AVERAGE\n";
1906 
1907                 break;
1908             }
1909 
1910 
1911             case (TileResource :: AVERAGE): {
1912                 resource_substring += "AVERAGE\n";
1913 
1914                 break;
1915             }
1916 
1917 
1918             case (TileResource :: ABOVE_AVERAGE): {
1919                 resource_substring += "ABOVE AVERAGE\n";
1920 
1921                 break;
1922             }
1923 
1924 
1925             case (TileResource :: GOOD): {
1926                 resource_substring += "GOOD\n";
1927 
1928                 break;
1929             }
1930 
1931 
1932             default: {
1933                 resource_substring += "???\n";
1934 
1935                 break;
1936             }
1937         }
1938     }
1939 
1940     else {
1941         resource_substring += "???\n";
1942     }
1943 
1944     return resource_substring;
1945 }   /* __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.

```
1828 {
1829     std::string type_substring = "TILE TYPE:         ";
1830 
1831     switch (this->tile_type) {
1832         case (TileType :: FOREST): {
1833             type_substring += "FOREST\n";
1834 
1835             break;
1836         }
1837 
1838 
1839         case (TileType :: LAKE): {
1840             type_substring += "LAKE\n";
1841 
1842             break;
1843         }
1844 
1845 
1846         case (TileType :: MOUNTAINS): {
1847             type_substring += "MOUNTAINS\n";
1848 
1849             break;
1850         }
1851 
1852 
1853         case (TileType :: OCEAN): {
1854             type_substring += "OCEAN\n";
1855 
1856             break;
1857         }
1858 
1859 
1860         case (TileType :: PLAINS): {
1861             type_substring += "PLAINS\n";
1862 
1863             break;
1864         }
1865 
1866 
1867         default: {
1868             type_substring += "???\n";
1869 
1870             break;
1871         }
1872     }
1873 
1874     return type_substring;
1875 }   /* __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             }
1274
1275             else {
1276                 this->__setIsSelected(false);
1277             }
1278
1279             break;
1280         }
1281
1282
1283         case (sf::Mouse::Right): {
1284             this->__setIsSelected(false);
1285
1286             break;
1287         }
1288
1289
1290         default: {
1291             // do nothing!
1292
1293             break;
1294         }
1295     }
1296
1297     return;
1298 }   /* __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.

```
1313 {
1314     if (this->build_menu_open) {
1315         return;
1316     }
1317 
1318     this->build_menu_open = true;
1319     this->assets_manager_ptr->getSound("build menu open")->play();
1320 
1321     return;
1322 }   /* __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.

```
1702 {
1703     //  1. implement key hold confirmation
1704     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1705         double colour_scalar =
1706             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1707 
1708         if (
1709             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1710         ) {
1711             this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1712                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1713             );
1714         }
1715 
1716         else {
1717             for (
1718                 size_t i = 0;
1719                 i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1720                 i++
1721             ) {
1722                 this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1723                     sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
```

```
1724                 );
1725             }
1726         }
1727
1728         this->scrap_improvement_frame += 4;
1729     }
1730
1731
1732     //  2. carry out scrapping
1733     else {
1734         this->draw_explosion = true;
1735         this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1736
1737         if (this->tile_improvement_ptr->production_menu_open) {
1738             this->tile_improvement_ptr->production_menu_open = false;
1739             this->assets_manager_ptr->getSound("build menu close")->play();
1740         }
1741
1742         delete this->tile_improvement_ptr;
1743         this->tile_improvement_ptr = NULL;
1744
1745         this->has_improvement = false;
1746
1747         this->scrap_improvement_frame = 0;
1748
1749         if (
1750             (this->tile_type == TileType :: LAKE) or
1751             (this->tile_type == TileType :: OCEAN)
1752         ) {
1753             this->decoration_cleared = false;
1754         }
1755
1756         this->__sendCreditsSpentMessage(SCRAP_COST);
1757         this->__sendTileStateMessage();
1758         this->__sendGameStateRequest();
1759     }
1760
1761     return;
1762 }   /* __scrapImprovement() */
```

4.7.3.21 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2139 {
2140     Message assess_neighbours_message;
2141
2142     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2143     assess_neighbours_message.subject = "assess neighbours";
2144
2145     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2146
2147     std::cout « "Assess neighbours message sent by " « this « std::endl;
2148
2149     return;
2150 }   /* __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.

```
2222 {
2223     Message credits_spent_message;
2224 
2225     credits_spent_message.channel = GAME_CHANNEL;
2226     credits_spent_message.subject = "credits spent";
2227 
2228     credits_spent_message.int_payload["credits spent"] = credits_spent;
2229 
2230     this->message_hub_ptr->sendMessage(credits_spent_message);
2231 
2232     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2233         << std::endl;
2234     return;
2235 }   /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2165 {
2166     Message game_state_request;
2167 
2168     game_state_request.channel = GAME_CHANNEL;
2169     game_state_request.subject = "state request";
2170 
2171     this->message_hub_ptr->sendMessage(game_state_request);
2172 
2173     std::cout << "Game state request message sent by " << this << std::endl;
2174     return;
2175 }   /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2250 {
2251     Message insufficient_credits_message;
2252 
2253     insufficient_credits_message.channel = GAME_CHANNEL;
2254     insufficient_credits_message.subject = "insufficient credits";
2255 
2256     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2257 
2258     std::cout << "Insufficient credits message sent by " << this << std::endl;
2259 
2260     return;
2261 }   /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1777 {
1778     Message tile_selected_message;
1779 
1780     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1781     tile_selected_message.subject = "tile selected";
1782 
1783     this->message_hub_ptr->sendMessage(tile_selected_message);
1784 
1785     return;
1786 }   /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2098 {
2099     Message tile_state_message;
2100 
2101     tile_state_message.channel = TILE_STATE_CHANNEL;
2102     tile_state_message.subject = "tile state";
2103 
2104 
2105     //                  32 char x 17 line console "--------------------------------\n";
2106     std::string console_string                    = "      **** TILE INFO ****       \n";
2107 
2108     console_string                               += this->__getTileCoordsSubstring();
2109     console_string                               += "                                \n";
2110 
2111     console_string                               += this->__getTileTypeSubstring();
2112     console_string                               += this->__getTileResourceSubstring();
2113     console_string                               += this->__getTileImprovementSubstring();
2114     console_string                               += "                                \n";
2115 
2116     console_string                               += this->__getTileOptionsSubstring();
2117 
2118     tile_state_message.string_payload["console string"] = console_string;
2119 
2120     this->message_hub_ptr->sendMessage(tile_state_message);
2121 
2122     std::cout « "Tile state message sent by " « this « std::endl;
2123     return;
2124 }   /* __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.

```
2192 {
2193     Message update_game_phase_message;
2194
2195     update_game_phase_message.channel = GAME_CHANNEL;
2196     update_game_phase_message.subject = "update game phase";
2197
2198     update_game_phase_message.string_payload["game phase"] = game_phase;
2199
2200     this->message_hub_ptr->sendMessage(update_game_phase_message);
2201
2202     std::cout « "Update game phase message sent by " « this « std::endl;
2203
2204     return;
2205 }   /* __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.

```
2684 {
2685     this->resource_assessed = true;
2686     this->resource_assessment = true;
2687 
2688     this->assets_manager_ptr->getSound("resource assessment")->play();
2689 
2690     this->__setResourceText();
2691     this->__sendTileStateMessage();
2692 
2693     return;
2694 }   /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2562 {
2563     switch (this->tile_type) {
2564         case (TileType :: FOREST): {
2565             this->tile_decoration_sprite.setTexture(
2566                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2567             );
2568 
2569             break;
2570         }
2571 
2572         case (TileType :: LAKE): {
2573             this->tile_decoration_sprite.setTexture(
2574                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2575             );
2576 
2577             break;
2578         }
2579 
2580         case (TileType :: MOUNTAINS): {
2581             this->tile_decoration_sprite.setTexture(
2582                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2583             );
2584 
2585             break;
2586         }
2587 
2588         case (TileType :: OCEAN): {
2589             this->tile_decoration_sprite.setTexture(
2590                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2591             );
2592 
2593             break;
2594         }
2595 
2596         case (TileType :: PLAINS): {
2597             this->tile_decoration_sprite.setTexture(
2598                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2599             );
2600 
2601             break;
2602         }
2603 
2604         default: {
2605             // do nothing!
2606 
2607             break;
2608         }
2609     }
2610 
2611 
2612     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```
2613         this->tile_decoration_sprite.setOrigin(
2614             this->tile_decoration_sprite.getLocalBounds().width / 2,
2615             this->tile_decoration_sprite.getLocalBounds().height / 2
2616         );
2617 
2618         this->tile_decoration_sprite.setPosition(
2619             this->position_x,
2620             this->position_y
2621         );
2622 
2623         if ((double)rand() / RAND_MAX > 0.5) {
2624             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2625         }
2626     }
2627 
2628     else {
2629         this->tile_decoration_sprite.setOrigin(
2630             this->tile_decoration_sprite.getLocalBounds().width / 2,
2631             this->tile_decoration_sprite.getLocalBounds().height
2632         );
2633 
2634         this->tile_decoration_sprite.setPosition(
2635             this->position_x,
2636             this->position_y + 12
2637         );
2638 
2639         if ((double)rand() / RAND_MAX > 0.5) {
2640             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2641         }
2642     }
2643 
2644     return;
2645 }   /* 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.

```
2825 {
2826     //  1. draw hex
2827     this->render_window_ptr->draw(this->tile_sprite);
2828 
2829     //  2. draw node
2830     if (this->show_node) {
2831         this->render_window_ptr->draw(this->node_sprite);
2832     }
2833 
2834     //  3. draw tile decoration
2835     if (not this->decoration_cleared) {
2836         this->render_window_ptr->draw(this->tile_decoration_sprite);
2837     }
2838 
2839     //  4. draw selection outline
2840     if (this->is_selected) {
2841         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2842 
2843         outline_colour.a =
2844             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2845 
2846         this->select_outline_sprite.setOutlineColor(outline_colour);
2847 
2848         this->render_window_ptr->draw(this->select_outline_sprite);
2849     }
2850 
2851     //  5. draw tile improvement
2852     if (this->has_improvement) {
2853         if (not this->tile_improvement_ptr->just_built) {
2854             this->tile_improvement_ptr->draw();
2855         }
2856     }
2857 
2858     //  6. draw resource
2859     if (this->show_resource) {
2860         this->render_window_ptr->draw(this->resource_chip_sprite);
2861         this->render_window_ptr->draw(this->resource_text);
2862     }
```

```
2863
2864     //  7. draw resource assessment notification
2865     if (this->resource_assessment) {
2866         int alpha = this->magnifying_glass_sprite.getColor().a;
2867
2868         alpha -= 0.05 * FRAMES_PER_SECOND;
2869         if (alpha < 0) {
2870             alpha = 0;
2871             this->resource_assessment = false;
2872         }
2873
2874         this->magnifying_glass_sprite.setColor(
2875             sf::Color(255, 255, 255, alpha)
2876         );
2877
2878         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2879     }
2880
2881     //  8. draw explosion, then settlement placement
2882     if (this->draw_explosion) {
2883         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2884
2885         if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2886             this->explosion_frame++;
2887         }
2888
2889         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2890             this->draw_explosion = false;
2891             this->explosion_frame = 0;
2892         }
2893     }
2894
2895     else if (this->has_improvement) {
2896         if (this->tile_improvement_ptr->just_built) {
2897             this->tile_improvement_ptr->draw();
2898         }
2899     }
2900
2901     //  9. build menu
2902     if (this->build_menu_open) {
2903         this->render_window_ptr->draw(this->build_menu_backing);
2904         this->render_window_ptr->draw(this->build_menu_backing_text);
2905
2906         for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2907             for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2908                 this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2909             }
2910             this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2911         }
2912     }
2913
2914     this->frame++;
2915     return;
2916 }   /* draw() */
```

4.7.3.47 processEvent()

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

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

```
2709 {
2710     //  1. process TileImprovement events
2711     if (
2712         this->is_selected and
2713         this->tile_improvement_ptr != NULL
2714     ) {
2715         this->tile_improvement_ptr->processEvent();
2716     }
2717
2718     //  2. process HexTile events
2719     if (this->event_ptr->type == sf::Event::KeyPressed) {
2720         this->__handleKeyPressEvents();
2721     }
2722
2723     if (this->event_ptr->type == sf::Event::KeyReleased) {
2724         this->__handleKeyReleaseEvents();
2725     }
```

```
2726
2727     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2728         this->__handleMouseButtonEvents();
2729     }
2730
2731     return;
2732 }    /* processEvent() */
```

4.7.3.48 processMessage()

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

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

```
2747 {
2748     //  1. process TileImprovement messages
2749     if (this->tile_improvement_ptr != NULL) {
2750         this->tile_improvement_ptr->processMessage();
2751     }
2752
2753     //  2. process HexTile messages
2754     if (this->is_selected) {
2755         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2756             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2757                 TILE_STATE_CHANNEL
2758             );
2759
2760             if (tile_state_message.subject == "state request") {
2761                 this->__sendTileStateMessage();
2762
2763                 std::cout « "Tile state request received by " « this « std::endl;
2764                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2765             }
2766         }
2767
2768         std::cout « "Current credits (HexTile): " « this->credits « " K" «
2769             std::endl;
2770     }
2771
2772     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2773         Message game_state_message = this->message_hub_ptr->receiveMessage(
2774             GAME_STATE_CHANNEL
2775         );
2776
2777         if (game_state_message.subject == "game state") {
2778             this->credits = game_state_message.int_payload["credits"];
2779             this->game_phase = game_state_message.string_payload["game phase"];
2780
2781             if (this->tile_improvement_ptr != NULL) {
2782                 this->tile_improvement_ptr->credits = this->credits;
2783                 this->tile_improvement_ptr->game_phase = this->game_phase;
2784
2785                 this->tile_improvement_ptr->month =
2786                     game_state_message.int_payload["month"];
2787
2788                 this->tile_improvement_ptr->demand_MWh =
2789                     game_state_message.int_payload["demand_MWh"];
2790
2791                 this->tile_improvement_ptr->demand_vec_MWh =
2792                     game_state_message.vector_payload["demand_vec_MWh"];
2793
2794                 this->tile_improvement_ptr->update();
2795             }
2796
2797             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
2798
2799             std::cout « "Game state message read and passed by " « this «
2800                 " (credits: " « this->credits « " K)" « std::endl;
2801
2802             if (this->is_selected) {
2803                 this->__sendTileStateMessage();
2804             }
2805         }
2806     }
2807
2808     return;
2809 }    /* 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].

```
2511 {
2512     //  1. check input
2513     if (input_value < 0 or input_value > 1) {
2514         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
2515         error_str += "not in the closed interval [0, 1]";
2516 
2517         #ifdef _WIN32
2518             std::cout « error_str « std::endl;
2519         #endif  /* _WIN32 */
2520 
2521         throw std::runtime_error(error_str);
2522     }
2523 
2524     //  2. convert input value to tile resource
2525     TileResource tile_resource;
2526 
2527     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2528         tile_resource = TileResource :: POOR;
2529     }
2530     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2531         tile_resource = TileResource :: BELOW_AVERAGE;
2532     }
2533     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2534         tile_resource = TileResource :: AVERAGE;
2535     }
2536     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2537         tile_resource = TileResource :: ABOVE_AVERAGE;
2538     }
2539     else {
2540         tile_resource = TileResource :: GOOD;
2541     }
2542 
2543     //  3. call alternate method
2544     this->setTileResource(tile_resource);
2545 
2546     return;
2547 }   /* 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.

```
2489 {
2490     this->tile_resource = tile_resource;
2491     this->__setResourceText();
2492 
2493     return;
2494 }   /* 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].

```
2439 {
2440     //  1. check input
2441     if (input_value < 0 or input_value > 1) {
2442         std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
2443         error_str += "not in the closed interval [0, 1]";
2444 
2445         #ifdef _WIN32
2446             std::cout « error_str « std::endl;
2447         #endif  /* _WIN32 */
2448 
2449         throw std::runtime_error(error_str);
2450     }
2451 
2452     //  2. convert input value to tile type
2453     TileType tile_type;
2454 
2455     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2456         tile_type = TileType :: LAKE;
2457     }
2458     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2459         tile_type = TileType :: PLAINS;
2460     }
2461     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2462         tile_type = TileType :: FOREST;
2463     }
2464     else {
2465         tile_type = TileType :: MOUNTAINS;
2466     }
2467 
2468     //  3. call alternate method
2469     this->setTileType(tile_type);
2470 
2471     return;
2472 }   /* 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.

```
2378 {
2379     this->tile_type = tile_type;
2380 
2381     switch (this->tile_type) {
2382         case (TileType :: FOREST): {
2383             this->tile_sprite.setFillColor(FOREST_GREEN);
2384 
2385             break;
2386         }
2387 
2388         case (TileType :: LAKE): {
```

```
2389             this->tile_sprite.setFillColor(LAKE_BLUE);
2390
2391             break;
2392         }
2393
2394         case (TileType :: MOUNTAINS): {
2395             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2396
2397             break;
2398         }
2399
2400         case (TileType :: OCEAN): {
2401             this->tile_sprite.setFillColor(OCEAN_BLUE);
2402
2403             break;
2404         }
2405
2406         case (TileType :: PLAINS): {
2407             this->tile_sprite.setFillColor(PLAINS_YELLOW);
2408
2409             break;
2410         }
2411
2412         default: {
2413             // do nothing!
2414
2415             break;
2416         }
2417     }
2418
2419     this->__setUpBuildMenu();
2420
2421     return;
2422 }   /* setTileType(TileType) */
```

4.7.3.53 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
2660 {
2661     if (this->show_resource) {
2662         this->show_resource = false;
2663     }
2664     else {
2665         this->show_resource = true;
2666     }
2667
2668     return;
2669 }   /* 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.
- unsigned int number_of_reads = 0

  The number of times the message has been read.
- 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 number_of_reads

```
unsigned int Message::number_of_reads = 0
```

The number of times the message has been read.

4.8.2.6 string_payload

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

A string payload.

4.8.2.7 subject

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

A string describing the message subject.

4.8.2.8 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 incrementMessageRead (std::string)

  Method to increment the number of times the first message in the channel has been read. 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.

```
526 {
527     this->clear();
528
529     std::cout « "MessageHub at " « this « " destroyed" « std::endl;
530
531     return;
532 }   /* ~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.

```
506 {
507 
508     this->clearMessages();
509     this->message_map.clear();
510 
511     return;
512 }   /* clear() */
```

4.9.3.3 clearMessages()

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

Method to clear messages from the MessageHub.

```
480 {
481     std::map<std::string, std::list<Message»::iterator map_iter;
482     for (
483         map_iter = this->message_map.begin();
484         map_iter != this->message_map.end();
485         map_iter++
486     ) {
487         map_iter->second.clear();
488     }
489 
490     return;
491 }   /* 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 incrementMessageRead()

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

Method to increment the number of times the first message in the channel has been read. 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.

```
385 {
386     //  1. check if channel is in map (if not, throw error)
387     if (this->message_map.count(channel) <= 0) {
388         std::string error_str = "ERROR  MessageHub::incrementMessageRead()  channel ";
389         error_str += channel;
390         error_str += " is not in message map";
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     //  2. check if channel is empty (if so, throw error)
400     if (this->message_map[channel].empty()) {
401         std::string error_str = "ERROR  MessageHub::incrementMessageRead()  channel ";
402         error_str += channel;
403         error_str += " is empty";
404 
405         #ifdef _WIN32
406             std::cout << error_str << std::endl;
407         #endif  /* _WIN32 */
408 
409         throw std::runtime_error(error_str);
410     }
411 
412     //  3. increment number of reads
413     this->message_map[channel].front().number_of_reads++;
414 
415     return;
416 }   /* incrementMessageRead( */
```

4.9.3.6 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() */
```

4.9.3.7 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.

```
434 {
435     //  1. check if channel is in map (if not, throw error)
436     if (this->message_map.count(channel) <= 0) {
437         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
438         error_str += channel;
439         error_str += " is not in message map";
440 
441         #ifdef _WIN32
442             std::cout « error_str « std::endl;
443         #endif  /* _WIN32 */
444 
445         throw std::runtime_error(error_str);
446     }
447 
448     //  2. check if channel is empty (if so, throw error)
449     if (this->message_map[channel].empty()) {
450         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
451         error_str += channel;
452         error_str += " is empty";
453 
454         #ifdef _WIN32
455             std::cout « error_str « std::endl;
456         #endif  /* _WIN32 */
457 
458         throw std::runtime_error(error_str);
459     }
460 
461     //  3. pop message
462     this->message_map[channel].pop_front();
463 
464     return;
465 }   /* popMessage() */
```

4.9.3.8 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() */
```

4.9.3.9 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() */
```

4.9.3.10 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() */
```

4.9.3.11 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:

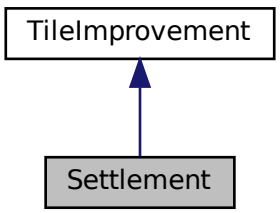
- header/ESC_core/MessageHub.h
- source/ESC_core/MessageHub.cpp

4.10 Settlement Class Reference

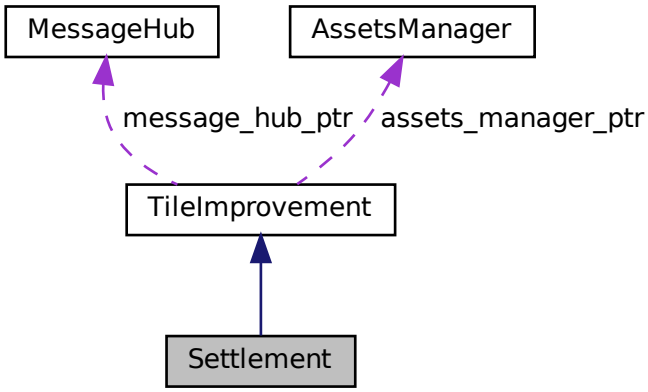
A settlement class (child class of TileImprovement).

```
#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:

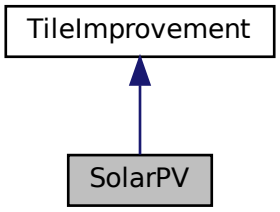
- header/Settlement.h
- source/Settlement.cpp

4.11 SolarPV Class Reference

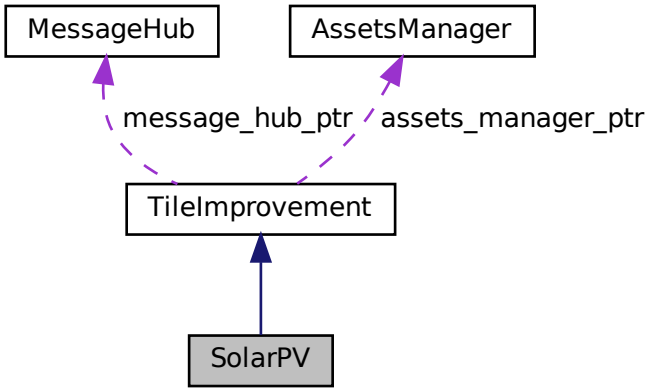
A settlement class (child class of TileImprovement).

```
#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 __repair (void)

  Helper method to repair the solar PV array.
- 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.

```
745   :
746 TileImprovement(
747     position_x,
748     position_y,
749     tile_resource,
750     event_ptr,
751     render_window_ptr,
752     assets_manager_ptr,
753     message_hub_ptr
754 )
755 {
756     //  1. set attributes
757
758     //  1.1. private
759     //...
760
761     //  1.2. public
762     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
763
764     this->is_running = false;
765
766     this->health = 100;
767
768     this->capacity_kW = 100;
769     this->upgrade_level = 1;
770
771     this->storage_kWh = 0;
772     this->storage_level = 0;
773
774     this->production_MWh = 0;
775     this->dispatch_MWh = 0;
776     this->dispatchable_MWh = 0;
777
778     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
779
780     this->capacity_factor_vec.resize(30, 0);
781     this->production_vec_MWh.resize(30, 0);
```

```
782     this->dispatch_vec_MWh.resize(30, 0);
783
784     this->tile_improvement_string = "SOLAR PV ARRAY";
785
786     this->__setUpTileImprovementSpriteStatic();
787     this->__computeCapacityFactors();
788     this->update();
789
790     std::cout « "SolarPV constructed at " « this « std::endl;
791
792     return;
793 }   /* SolarPV() */
```

4.11.2.2 ~SolarPV()

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

Destructor for the SolarPV class.

```
1125 {
1126     std::cout « "SolarPV at " « this « " destroyed" « std::endl;
1127
1128     return;
1129 }   /* ~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.

```
290 {
291     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
292     std::default_random_engine generator(seed);
293 
294     double mean =
295         this->tile_resource_scalar * MEAN_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
296 
297     double stdev = STDEV_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
298 
299     if (this->tile_resource_scalar > 1) {
300         stdev /= this->tile_resource_scalar;
301     }
302 
303     std::normal_distribution<double> normal_dist(mean, stdev);
304 
305     double capacity_factor = 0;
306 
307     for (int i = 0; i < 30; i++) {
308         capacity_factor = normal_dist(generator);
309 
310         if (capacity_factor < 0) {
311             capacity_factor = 0;
312         }
313 
314         this->capacity_factor_vec[i] = capacity_factor;
315     }
316 
317     return;
318 }   /* __computeCapacityFactors() */
```

4.11.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
361 {
362     double stored_energy_MWh = 0;
363     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
364 
365     double demand_MWh = 0;
366     double production_MWh = 0;
367     double dispatchable_MWh = 0;
368     double difference_MWh = 0;
369 
370     double room_MWh = 0;
371 
372     for (int i = 0; i < 30; i++) {
373         demand_MWh = this->demand_vec_MWh[i];
374         production_MWh = this->production_vec_MWh[i];
375 
376         if (production_MWh <= demand_MWh) {
377             this->dispatch_vec_MWh[i] = production_MWh;
378             dispatchable_MWh += this->dispatch_vec_MWh[i];
379 
380             difference_MWh = demand_MWh - production_MWh;
381 
382             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
383                 if (difference_MWh > stored_energy_MWh) {
384                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
385                     dispatchable_MWh += stored_energy_MWh;
386                     stored_energy_MWh = 0;
387                 }
388 
389                 else {
390                     this->dispatch_vec_MWh[i] += difference_MWh;
391                     dispatchable_MWh += difference_MWh;
392                     stored_energy_MWh -= difference_MWh;
393                 }
```

```
394             }
395         }
396
397         else {
398             this->dispatch_vec_MWh[i] = demand_MWh;
399             dispatchable_MWh += this->dispatch_vec_MWh[i];
400
401             difference_MWh = production_MWh - demand_MWh;
402
403             if (
404                 (storage_capacity_MWh > 0) and
405                 (stored_energy_MWh < storage_capacity_MWh)
406             ) {
407                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
408
409                 if (difference_MWh > room_MWh) {
410                     stored_energy_MWh += room_MWh;
411                 }
412
413                 else {
414                     stored_energy_MWh += difference_MWh;
415                 }
416             }
417         }
418     }
419
420     this->dispatchable_MWh = round(dispatchable_MWh);
421
422     if (this->dispatch_MWh != this->dispatchable_MWh) {
423         this->dispatch_MWh = this->dispatchable_MWh;
424     }
425
426     return;
427 }   /* __computeDispatch() */
```

4.11.3.4 __computeProduction()

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

Helper method to compute production values.

```
333 {
334     double production_MWh = 0;
335
336     for (int i = 0; i < 30; i++) {
337         this->production_vec_MWh[i] =
338             this->max_daily_production_MWh * this->capacity_factor_vec[i];
339
340         production_MWh += this->production_vec_MWh[i];
341     }
342
343     this->production_MWh = round(production_MWh);
344
345     return;
346 }   /* __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.

```
568 {
569     //  1. draw power capacity upgrade sprite
570     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
571     this->tile_improvement_sprite_static.setPosition(400 - 100, 400 - 32);
572 
573     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
574     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
575 
576     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();
577     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
578 
579     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
580 
581     this->tile_improvement_sprite_static.setPosition(initial_position);
582     this->tile_improvement_sprite_static.setColor(initial_colour);
```

```
583     this->tile_improvement_sprite_static.setScale(initial_scale);
584
585     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
586
587
588     //  2. draw power capacity upgrade text
589     //                  16 char line = "                \n"
590     std::string power_upgrade_string = "POWER CAPACITY  \n";
591     power_upgrade_string            += "                \n";
592
593     power_upgrade_string            += "CAPACITY:  ";
594     power_upgrade_string            += std::to_string(this->capacity_kW);
595     power_upgrade_string            += " kW\n";
596
597     power_upgrade_string            += "LEVEL:     ";
598     power_upgrade_string            += std::to_string(this->upgrade_level);
599     power_upgrade_string            += "\n\n";
600
601     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
602         power_upgrade_string        += "[W]: + 100 kW (";
603         power_upgrade_string        +=  std::to_string(SOLAR_PV_BUILD_COST);
604         power_upgrade_string        += " K)\n";
605     }
606
607     else {
608         power_upgrade_string        += " * MAX LEVEL *  \n";
609     }
610
611     sf::Text power_upgrade_text = sf::Text(
612         power_upgrade_string,
613         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
614         16
615     );
616
617     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
618     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
619     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
620
621     this->render_window_ptr->draw(power_upgrade_text);
622
623
624     //  3. draw energy capacity (storage) upgrade sprite
625     this->render_window_ptr->draw(this->storage_upgrade_sprite);
626     this->render_window_ptr->draw(this->upgrade_plus_sprite);
627
628
629     //  4. draw energy capacity (storage) upgrade text
630     //                   16 char line = "                \n"
631     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
632     energy_upgrade_string            += "                \n";
633
634     energy_upgrade_string            += "CAPACITY:  ";
635     energy_upgrade_string            += std::to_string(this->storage_level * 200);
636     energy_upgrade_string            += " kWh\n";
637
638     energy_upgrade_string            += "LEVEL:     ";
639     energy_upgrade_string            += std::to_string(this->storage_level);
640     energy_upgrade_string            += "\n\n";
641
642     if (this->storage_level < MAX_STORAGE_LEVELS) {
643         energy_upgrade_string        += "[D]: + 200 kWh (";
644         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
645         energy_upgrade_string        += " K)\n";
646     }
647
648     else {
649         energy_upgrade_string += " * MAX LEVEL *  \n";
650     }
651
652     sf::Text energy_upgrade_text = sf::Text(
653         energy_upgrade_string,
654         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
655         16
656     );
657
658     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
659     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
660     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
661
662     this->render_window_ptr->draw(energy_upgrade_text);
663
664     return;
665 }   /* __drawUpgradeOptions() */
```

4.11.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
442 {
443     if (this->just_built) {
444         return;
445     }
446 
447     switch (this->event_ptr->key.code) {
448         case (sf::Keyboard::U): {
449             this->__openUpgradeMenu();
450 
451             break;
452         }
453 
454 
455         case (sf::Keyboard::W): {
456             if (this->production_menu_open) {
457                 this->dispatch_MWh++;
458 
459                 if (this->dispatch_MWh > this->dispatchable_MWh) {
460                     this->dispatch_MWh = 0;
461                 }
462 
463                 this->__computeProductionCosts();
464                 this->assets_manager_ptr->getSound("interface click")->play();
465             }
466 
467             else if (this->upgrade_menu_open) {
468                 this->__upgradePowerCapacity();
469             }
470 
471             break;
472         }
473 
474 
475         case (sf::Keyboard::S): {
476             if (this->production_menu_open) {
477                 this->dispatch_MWh--;
478 
479                 if (this->dispatch_MWh < 0) {
480                     this->dispatch_MWh = this->dispatchable_MWh;
481                 }
482 
483                 this->__computeProductionCosts();
484                 this->assets_manager_ptr->getSound("interface click")->play();
485             }
486 
487             break;
488         }
489 
490 
491         case (sf::Keyboard::D): {
492             if (this->upgrade_menu_open) {
493                 this->__upgradeStorageCapacity();
494                 this->__computeProduction();
495                 this->__computeDispatch();
496             }
497 
498             break;
499         }
500 
501 
502         default: {
503             // do nothing!
504 
505             break;
506         }
507     }
508 
509     return;
510 }   /* __handleKeyPressEvents() */
```

4.11.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
525 {
526     if (this->just_built) {
527         return;
528     }
529 
530     switch (this->event_ptr->mouseButton.button) {
531         case (sf::Mouse::Left): {
532             //...
533 
534             break;
535         }
536 
537 
538         case (sf::Mouse::Right): {
539             //...
540 
541             break;
542         }
543 
544 
545         default: {
546             // do nothing!
547 
548             break;
549         }
550     }
551 
552     return;
553 }   /* __handleMouseButtonEvents() */
```

4.11.3.10 __repair()

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

Helper method to repair the solar PV array.

Reimplemented from TileImprovement.

```
257 {
258     if (this->credits < SOLAR_PV_BUILD_COST) {
259         std::cout « "Cannot repair solar PV: insufficient credits (need "
260             « SOLAR_PV_BUILD_COST « " K)" « std::endl;
261 
262         this->__sendInsufficientCreditsMessage();
263         return;
264     }
265 
266     TileImprovement :: __repair();
267 
268     this->just_upgraded = true;
269 
270     this->__sendCreditsSpentMessage(SOLAR_PV_BUILD_COST);
271     this->__sendTileStateRequest();
272     this->__sendGameStateRequest();
273 
274     return;
275 }   /* __repair() */
```

4.11.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
680 {
681     Message improvement_state_message;
682 
```

```
683     improvement_state_message.channel = GAME_CHANNEL;
684     improvement_state_message.subject = "improvement state";
685
686     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
687     improvement_state_message.int_payload["operation_maintenance_cost"] =
688         this->operation_maintenance_cost;
689
690     this->message_hub_ptr->sendMessage(improvement_state_message);
691
692     std::cout « "Improvement state message sent by " « this « std::endl;
693
694     return;
695 }   /* __sendImprovementStateMessage() */
```

4.11.3.12 __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() */
```

4.11.3.13 __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     TileImprovement :: __repair();
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() */
```

4.11.3.14 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
898 {
899     //  1. send improvement state message
900     this->__sendImprovementStateMessage();
901
902     //  2. update
903     this->__computeCapacityFactors();
904     this->update();
905
906     //  3. handle start/stop
907     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
908         this->is_running = true;
909     }
910
911     else if (this->is_running and (this->dispatch_MWh <= 0)) {
912         this->is_running = false;
913     }
914
915     //  4. handle equipment health
916     if (this->is_running) {
917         this->health--;
918
919         if (this->health <= 0) {
920             this->__breakdown();
921         }
922     }
923
924     //  5. send tile state request (if selected)
925     if (this->is_selected) {
926         this->__sendTileStateRequest();
927     }
928
929     return;
930 }   /* advanceTurn() */
```

4.11.3.15 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.

```
1019 {
1020     //  1. if just built, call base method and return
1021     if (this->just_built) {
1022         TileImprovement :: draw();
```

```
1023
1024            return;
1025        }
1026
1027
1028        //  2. handle upgrade effects
1029        if (this->just_upgraded) {
1030            this->tile_improvement_sprite_static.setColor(
1031                sf::Color(
1032                    255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1033                    255,
1034                    255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1035                    255
1036                )
1037            );
1038
1039            this->tile_improvement_sprite_static.setScale(
1040                sf::Vector2f(
1041                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1042                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1043                )
1044            );
1045
1046            this->upgrade_frame++;
1047        }
1048
1049        if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1050            this->tile_improvement_sprite_static.setColor(
1051                sf::Color(255,255,255,255)
1052            );
1053
1054            this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
1055
1056            this->just_upgraded = false;
1057            this->upgrade_frame = 0;
1058        }
1059
1060
1061        //  3. draw static sprite
1062        this->render_window_ptr->draw(this->tile_improvement_sprite_static);
1063
1064
1065        //  4. draw storage upgrades
1066        for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1067            this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1068        }
1069
1070
1071        //  5. handle dispatch illustration
1072        if (this->dispatch_MWh > 0) {
1073            this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1074            this->__drawDispatch();
1075        }
1076
1077
1078        //  6. draw production menu
1079        if (this->production_menu_open) {
1080            this->render_window_ptr->draw(this->production_menu_backing);
1081            this->render_window_ptr->draw(this->production_menu_backing_text);
1082
1083            this->__drawProductionMenu();
1084        }
1085
1086
1087        //  7. draw upgrade menu
1088        if (this->upgrade_menu_open) {
1089            this->render_window_ptr->draw(this->upgrade_menu_backing);
1090            this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1091
1092            this->__drawUpgradeOptions();
1093        }
1094
1095
1096        //  10. handle broken effects
1097        if (this->is_broken) {
1098            this->tile_improvement_sprite_static.setColor(
1099                sf::Color(
1100                    255,
1101                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1102                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1103                    255
1104                )
1105            );
1106        }
1107
1108        this->frame++;
1109        return;
```

```
1110 }   /* draw() */
```

4.11.3.16 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
810 {
811     //                  32 char x 17 line console "--------------------------------\n";
812     std::string options_substring                = "CAPACITY:      ";
813     options_substring                           += std::to_string(this->capacity_kW);
814     options_substring                           += " kW (level ";
815     options_substring                           += std::to_string(this->upgrade_level);
816     options_substring                           += ")\n";
817
818     options_substring                           += "PRODUCTION:    ";
819     options_substring                           += std::to_string(this->production_MWh);
820     options_substring                           += " MWh\n";
821
822     options_substring                           += "DISPATCHABLE:  ";
823     options_substring                           += std::to_string(this->dispatchable_MWh);
824     options_substring                           += " MWh\n";
825
826     options_substring                           += "HEALTH:      ";
827     options_substring                           += std::to_string(this->health);
828     options_substring                           += "/100";
829
830     if (this->health <= 0) {
831         options_substring                       += " ** BROKEN! **\n";
832     }
833
834     else {
835         options_substring                       += "\n";
836     }
837
838     options_substring                           += "                                \n";
839     options_substring                           += "   **** SOLAR PV OPTIONS ****   \n";
840     options_substring                           += "                                \n";
841
842     if (this->is_broken) {
843         options_substring                       += "     [R]:  REPAIR (";
844         options_substring                       += std::to_string(SOLAR_PV_BUILD_COST);
845         options_substring                       += " K)\n";
846     }
847
848     else {
849         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
850     }
851
852     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
853     options_substring                           += "HOLD [P]:  SCRAP (";
854     options_substring                           += std::to_string(SCRAP_COST);
855     options_substring                           += " K)";
856
857     return options_substring;
858 }   /* getTileOptionsSubstring() */
```

4.11.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
970 {
971     TileImprovement :: processEvent();
972 
973     if (this->event_ptr->type == sf::Event::KeyPressed) {
974         this->__handleKeyPressEvents();
975     }
976 
977     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
978         this->__handleMouseButtonEvents();
979     }
980 
981     return;
982 }   /* processEvent() */
```

4.11.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
997 {
998     TileImprovement :: processMessage();
999 
1000     //...
1001 
1002     return;
1003 }   /* processMessage() */
```

4.11.3.19 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.

```
875 {
876     TileImprovement :: setIsSelected(is_selected);
877 
878     if (this->is_running and this->is_selected) {
879         this->assets_manager_ptr->getSound("solar hum")->play();
880     }
881 
```

```
882     return;
883 }   /* setIsSelected() */
```

4.11.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
945 {
946     this->__computeProduction();
947     this->__computeProductionCosts();
948     this->__computeDispatch();
949 
950     if (this->is_selected) {
951         this->__sendTileStateRequest();
952     }
953 
954     return;
955 }   /* 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:

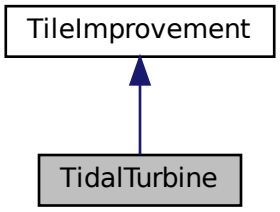
- header/SolarPV.h
- source/SolarPV.cpp

4.12 TidalTurbine Class Reference

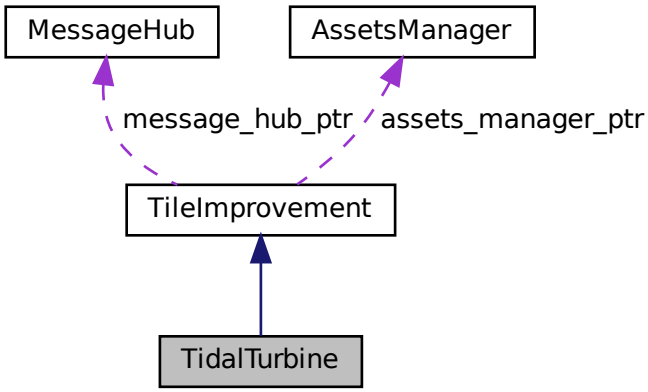
A settlement class (child class of TileImprovement).

`#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.
- double bobbing_y

  The bobbing extent of the tidal turbine.
- 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 __repair (void)

  Helper method to repair the tidal turbine.

- 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.

```
747   :
748 TileImprovement(
749     position_x,
750     position_y,
751     tile_resource,
752     event_ptr,
753     render_window_ptr,
754     assets_manager_ptr,
755     message_hub_ptr
756 )
757 {
758     //  1. set attributes
759
760     //  1.1. private
761     //...
762
763     //  1.2. public
764     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
765
766     this->is_running = false;
767
768     this->health = 100;
769
770     this->capacity_kW = 100;
771     this->upgrade_level = 1;
772
773     this->storage_kWh = 0;
774     this->storage_level = 0;
775
776     this->production_MWh = 0;
777     this->dispatch_MWh = 0;
778     this->dispatchable_MWh = 0;
779
780     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
781
782     this->rotor_drotation = 64 * SECONDS_PER_FRAME;
783     this->bobbing_y = 4;
784
785     this->capacity_factor_vec.resize(30, 0);
786     this->production_vec_MWh.resize(30, 0);
787     this->dispatch_vec_MWh.resize(30, 0);
788
789     this->tile_improvement_string = "TIDAL TURBINE";
790
791     this->__setUpTileImprovementSpriteAnimated();
792     this->__computeCapacityFactors();
793     this->update();
794
795     std::cout « "TidalTurbine constructed at " « this « std::endl;
796
797     return;
798 }   /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

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

Destructor for the TidalTurbine class.

```
1156 {
1157     std::cout « "TidalTurbine at " « this « " destroyed" « std::endl;
1158
1159     return;
1160 }   /* ~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.

```
307 {
308     for (int i = 0; i < 30; i++) {
309         this->capacity_factor_vec[i] =
310             this->tile_resource_scalar * DAILY_TIDAL_CAPACITY_FACTOR;
311     }
312 
313     return;
314 }   /* __computeCapacityFactors() */
```

4.12.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
357 {
358     double stored_energy_MWh = 0;
359     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
360 
361     double demand_MWh = 0;
362     double production_MWh = 0;
363     double dispatchable_MWh = 0;
364     double difference_MWh = 0;
365 
366     double room_MWh = 0;
367 
368     for (int i = 0; i < 30; i++) {
369         demand_MWh = this->demand_vec_MWh[i];
370         production_MWh = this->production_vec_MWh[i];
371 
372         if (production_MWh <= demand_MWh) {
373             this->dispatch_vec_MWh[i] = production_MWh;
374             dispatchable_MWh += this->dispatch_vec_MWh[i];
375 
376             difference_MWh = demand_MWh - production_MWh;
377 
378             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
379                 if (difference_MWh > stored_energy_MWh) {
380                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
381                     dispatchable_MWh += stored_energy_MWh;
382                     stored_energy_MWh = 0;
383                 }
384 
```

```
385                 else {
386                     this->dispatch_vec_MWh[i] += difference_MWh;
387                     dispatchable_MWh += difference_MWh;
388                     stored_energy_MWh -= difference_MWh;
389                 }
390             }
391         }
392 
393         else {
394             this->dispatch_vec_MWh[i] = demand_MWh;
395             dispatchable_MWh += this->dispatch_vec_MWh[i];
396 
397             difference_MWh = production_MWh - demand_MWh;
398 
399             if (
400                 (storage_capacity_MWh > 0) and
401                 (stored_energy_MWh < storage_capacity_MWh)
402             ) {
403                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
404 
405                 if (difference_MWh > room_MWh) {
406                     stored_energy_MWh += room_MWh;
407                 }
408 
409                 else {
410                     stored_energy_MWh += difference_MWh;
411                 }
412             }
413         }
414     }
415 
416     this->dispatchable_MWh = round(dispatchable_MWh);
417 
418     if (this->dispatch_MWh != this->dispatchable_MWh) {
419         this->dispatch_MWh = this->dispatchable_MWh;
420     }
421 
422     return;
423 }   /* __computeDispatch() */
```

4.12.3.4 __computeProduction()

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

Helper method to compute production values.

```
329 {
330     double production_MWh = 0;
331 
332     for (int i = 0; i < 30; i++) {
333         this->production_vec_MWh[i] =
334             this->max_daily_production_MWh * this->capacity_factor_vec[i];
335 
336         production_MWh += this->production_vec_MWh[i];
337     }
338 
339     this->production_MWh = round(production_MWh);
340 
341     return;
342 }   /* __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.

```
564 {
565     //  1. draw power capacity upgrade sprite
566     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
567         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
568         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
569 
570         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
571         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
572 
```

```
573         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
574         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
575 
576         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
577         this->tile_improvement_sprite_animated[i].setRotation(0);
578 
579         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
580 
581         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
582         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
583         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
584         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
585     }
586 
587     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
588 
589 
590     //  2. draw power capacity upgrade text
591     //                  16 char line = "                \n"
592     std::string power_upgrade_string = "POWER CAPACITY  \n";
593     power_upgrade_string            += "                \n";
594 
595     power_upgrade_string            += "CAPACITY:  ";
596     power_upgrade_string            += std::to_string(this->capacity_kW);
597     power_upgrade_string            += " kW\n";
598 
599     power_upgrade_string            += "LEVEL:     ";
600     power_upgrade_string            += std::to_string(this->upgrade_level);
601     power_upgrade_string            += "\n\n";
602 
603     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
604         power_upgrade_string        += "[W]: + 100 kW (";
605         power_upgrade_string        +=  std::to_string(TIDAL_TURBINE_BUILD_COST);
606         power_upgrade_string        += " K)\n";
607     }
608 
609     else {
610         power_upgrade_string        += " * MAX LEVEL *  \n";
611     }
612 
613     sf::Text power_upgrade_text = sf::Text(
614         power_upgrade_string,
615         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
616         16
617     );
618 
619     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
620     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
621     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
622 
623     this->render_window_ptr->draw(power_upgrade_text);
624 
625 
626     //  3. draw energy capacity (storage) upgrade sprite
627     this->render_window_ptr->draw(this->storage_upgrade_sprite);
628     this->render_window_ptr->draw(this->upgrade_plus_sprite);
629 
630 
631     //  4. draw energy capacity (storage) upgrade text
632     //                   16 char line = "                \n"
633     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
634     energy_upgrade_string            += "                \n";
635 
636     energy_upgrade_string            += "CAPACITY:  ";
637     energy_upgrade_string            += std::to_string(this->storage_level * 200);
638     energy_upgrade_string            += " kWh\n";
639 
640     energy_upgrade_string            += "LEVEL:     ";
641     energy_upgrade_string            += std::to_string(this->storage_level);
642     energy_upgrade_string            += "\n\n";
643 
644     if (this->storage_level < MAX_STORAGE_LEVELS) {
645         energy_upgrade_string        += "[D]: + 200 kWh (";
646         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
647         energy_upgrade_string        += " K)\n";
648     }
649 
650     else {
651         energy_upgrade_string += " * MAX LEVEL *  \n";
652     }
653 
654     sf::Text energy_upgrade_text = sf::Text(
655         energy_upgrade_string,
656         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
657         16
658     );
659 
```

```
660     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
661     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
662     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
663 
664     this->render_window_ptr->draw(energy_upgrade_text);
665 
666     return;
667 }   /* __drawUpgradeOptions() */
```

4.12.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
438 {
439     if (this->just_built) {
440         return;
441     }
442 
443     switch (this->event_ptr->key.code) {
444         case (sf::Keyboard::U): {
445             this->__openUpgradeMenu();
446 
447             break;
448         }
449 
450 
451         case (sf::Keyboard::W): {
452             if (this->production_menu_open) {
453                 this->dispatch_MWh++;
454 
455                 if (this->dispatch_MWh > this->dispatchable_MWh) {
456                     this->dispatch_MWh = 0;
457                 }
458 
459                 this->__computeProductionCosts();
460                 this->assets_manager_ptr->getSound("interface click")->play();
461             }
462 
463             else if (this->upgrade_menu_open) {
464                 this->__upgradePowerCapacity();
465             }
466 
467             break;
468         }
469 
470 
471         case (sf::Keyboard::S): {
472             if (this->production_menu_open) {
473                 this->dispatch_MWh--;
474 
475                 if (this->dispatch_MWh < 0) {
476                     this->dispatch_MWh = this->dispatchable_MWh;
477                 }
478 
479                 this->__computeProductionCosts();
480                 this->assets_manager_ptr->getSound("interface click")->play();
481             }
482 
483             break;
484         }
485 
486 
487         case (sf::Keyboard::D): {
488             if (this->upgrade_menu_open) {
489                 this->__upgradeStorageCapacity();
490                 this->__computeProduction();
491                 this->__computeDispatch();
492             }
493 
494             break;
495         }
496 
497 
498         default: {
499             // do nothing!
500 
```

```
501             break;
502         }
503     }
504
505     return;
506 }   /* __handleKeyPressEvents() */
```

4.12.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
521 {
522     if (this->just_built) {
523         return;
524     }
525
526     switch (this->event_ptr->mouseButton.button) {
527         case (sf::Mouse::Left): {
528             //...
529
530             break;
531         }
532
533
534         case (sf::Mouse::Right): {
535             //...
536
537             break;
538         }
539
540
541         default: {
542             // do nothing!
543
544             break;
545         }
546     }
547
548     return;
549 }   /* __handleMouseButtonEvents() */
```

4.12.3.10 __repair()

```
void TidalTurbine::__repair (
            void  )  [private], [virtual]
```

Helper method to repair the tidal turbine.

Reimplemented from TileImprovement.

```
274 {
275     if (this->credits < TIDAL_TURBINE_BUILD_COST) {
276         std::cout « "Cannot repair tidal turbine: insufficient credits (need "
277             « TIDAL_TURBINE_BUILD_COST « " K)" « std::endl;
278
279         this->__sendInsufficientCreditsMessage();
280         return;
281     }
282
283     TileImprovement :: __repair();
284
285     this->just_upgraded = true;
286
287     this->__sendCreditsSpentMessage(TIDAL_TURBINE_BUILD_COST);
288     this->__sendTileStateRequest();
289     this->__sendGameStateRequest();
290
291     return;
292 }   /* __repair() */
```

4.12.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
682 {
683     Message improvement_state_message;
684 
685     improvement_state_message.channel = GAME_CHANNEL;
686     improvement_state_message.subject = "improvement state";
687 
688     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
689     improvement_state_message.int_payload["operation_maintenance_cost"] =
690         this->operation_maintenance_cost;
691 
692     this->message_hub_ptr->sendMessage(improvement_state_message);
693 
694     std::cout << "Improvement state message sent by " << this << std::endl;
695 
696     return;
697 }   /* __sendImprovementStateMessage() */
```

4.12.3.12 __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() */
```

4.12.3.13 __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     TileImprovement :: __repair();
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() */
```

4.12.3.14 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
904 {
905     //  1. send improvement state message
906     this->__sendImprovementStateMessage();
907 
908     //  2. update
909     this->__computeCapacityFactors();
910     this->update();
911 
912     //  3. handle start/stop
913     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
914         this->is_running = true;
915     }
916 
917     else if (this->is_running and (this->dispatch_MWh <= 0)) {
918         this->is_running = false;
919     }
920 
921     //  4. handle equipment health
922     if (this->is_running) {
923         this->health--;
924 
925         if (this->health <= 0) {
926             this->__breakdown();
927         }
928     }
929 
```

```
930     //  5. send tile state request (if selected)
931     if (this->is_selected) {
932         this->__sendTileStateRequest();
933     }
934 
935     return;
936 }   /* advanceTurn() */
```

4.12.3.15 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.

```
1025 {
1026     //  1. if just built, call base method and return
1027     if (this->just_built) {
1028         TileImprovement :: draw();
1029 
1030         return;
1031     }
1032 
1033 
1034     //  2. handle upgrade effects
1035     if (this->just_upgraded) {
1036         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1037             this->tile_improvement_sprite_animated[i].setColor(
1038                 sf::Color(
1039                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1040                     255,
1041                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1042                     255
1043                 )
1044             );
1045 
1046             this->tile_improvement_sprite_animated[i].setScale(
1047                 sf::Vector2f(
1048                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1049                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1050                 )
1051             );
1052         }
1053 
1054         this->upgrade_frame++;
1055     }
1056 
1057     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1058         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1059             this->tile_improvement_sprite_animated[i].setColor(
1060                 sf::Color(255,255,255,255)
1061             );
1062 
1063             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1064         }
1065 
1066         this->just_upgraded = false;
1067         this->upgrade_frame = 0;
1068     }
1069 
1070 
1071     //  3. handle bobbing
1072     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1073         this->tile_improvement_sprite_animated[i].setPosition(
1074             this->position_x,
1075             this->position_y + this->bobbing_y * cos(
1076                 (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1077             )
1078         );
1079     }
1080 
1081 
1082     //  4. draw first element of animated sprite
1083     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1084 
1085 
```

```
1086     //  5. draw second element of animated sprite
1087     if (this->is_running) {
1088         this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1089     }
1090 
1091     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1092 
1093 
1094     //  6. draw storage upgrades
1095     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1096         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1097     }
1098 
1099 
1100     //  7. handle dispatch illustration
1101     if (this->dispatch_MWh > 0) {
1102         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1103         this->__drawDispatch();
1104     }
1105 
1106 
1107     //  8. draw production menu
1108     if (this->production_menu_open) {
1109         this->render_window_ptr->draw(this->production_menu_backing);
1110         this->render_window_ptr->draw(this->production_menu_backing_text);
1111 
1112         this->__drawProductionMenu();
1113     }
1114 
1115 
1116     //  9. draw upgrade menu
1117     if (this->upgrade_menu_open) {
1118         this->render_window_ptr->draw(this->upgrade_menu_backing);
1119         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1120 
1121         this->__drawUpgradeOptions();
1122     }
1123 
1124 
1125     //  10. handle broken effects
1126     if (this->is_broken) {
1127         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1128             this->tile_improvement_sprite_animated[i].setColor(
1129                 sf::Color(
1130                     255,
1131                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1132                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1133                     255
1134                 )
1135             );
1136         }
1137     }
1138 
1139     this->frame++;
1140     return;
1141 }   /* draw() */
```

4.12.3.16 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
815 {
816     //                  32 char x 17 line console "--------------------------------\n";
817     std::string options_substring                = "CAPACITY:      ";
818     options_substring                           += std::to_string(this->capacity_kW);
819     options_substring                           += " kW (level ";
```

```
820     options_substring                       += std::to_string(this->upgrade_level);
821     options_substring                       += ")\n";
822
823     options_substring                       += "PRODUCTION:    ";
824     options_substring                       += std::to_string(this->production_MWh);
825     options_substring                       += " MWh\n";
826
827     options_substring                       += "DISPATCHABLE:  ";
828     options_substring                       += std::to_string(this->dispatchable_MWh);
829     options_substring                       += " MWh\n";
830
831     options_substring                       += "HEALTH:      ";
832     options_substring                       += std::to_string(this->health);
833     options_substring                       += "/100";
834
835     if (this->health <= 0) {
836         options_substring                   += " ** BROKEN! **\n";
837     }
838
839     else {
840         options_substring                   += "\n";
841     }
842
843     options_substring                       += "                                \n";
844     options_substring                       += "**** TIDAL TURBINE OPTIONS **** \n";
845     options_substring                       += "                                \n";
846
847     if (this->is_broken) {
848         options_substring                   += "     [R]:  REPAIR (";
849         options_substring                   += std::to_string(TIDAL_TURBINE_BUILD_COST);
850         options_substring                   += " K)\n";
851     }
852
853     else {
854         options_substring                   += "     [E]:  OPEN PRODUCTION MENU \n";
855     }
856
857     options_substring                       += "     [U]:  OPEN UPGRADE MENU    \n";
858     options_substring                       += "HOLD [P]:  SCRAP (";
859     options_substring                       += std::to_string(SCRAP_COST);
860     options_substring                       += " K)";
861
862     return options_substring;
863 }   /* getTileOptionsSubstring() */
```

4.12.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
976 {
977     TileImprovement :: processEvent();
978
979     if (this->event_ptr->type == sf::Event::KeyPressed) {
980         this->__handleKeyPressEvents();
981     }
982
983     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
984         this->__handleMouseButtonEvents();
985     }
986
987     return;
988 }   /* processEvent() */
```

4.12.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1003 {
1004     TileImprovement :: processMessage();
1005
1006     //...
1007
1008     return;
1009 }   /* processMessage() */
```

4.12.3.19 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.

```
880 {
881     TileImprovement :: setIsSelected(is_selected);
882
883     if (this->is_running and this->is_selected) {
884         this->assets_manager_ptr->getSound("water flow")->play();
885     }
886
887     return;
888 }   /* setIsSelected() */
```

4.12.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
951 {
952     this->__computeProduction();
953     this->__computeProductionCosts();
954     this->__computeDispatch();
955
956     if (this->is_selected) {
957         this->__sendTileStateRequest();
958     }
959
960     return;
961 }   /* update() */
```

4.12.4 Member Data Documentation

4.12.4.1 bobbing_y

```
double TidalTurbine::bobbing_y
```

The bobbing extent of the tidal turbine.

4.12.4.2 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.12.4.3 capacity_kW

```
int TidalTurbine::capacity_kW
```

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

4.12.4.4 dispatch_MWh

```
int TidalTurbine::dispatch_MWh
```

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

4.12.4.5 dispatch_vec_MWh

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

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

4.12.4.6 dispatchable_MWh

```
int TidalTurbine::dispatchable_MWh
```

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

4.12.4.7 max_daily_production_MWh

```
double TidalTurbine::max_daily_production_MWh
```

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

4.12.4.8 production_MWh

```
int TidalTurbine::production_MWh
```

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

4.12.4.9 production_vec_MWh

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

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

4.12.4.10 rotor_drotation

```
double TidalTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

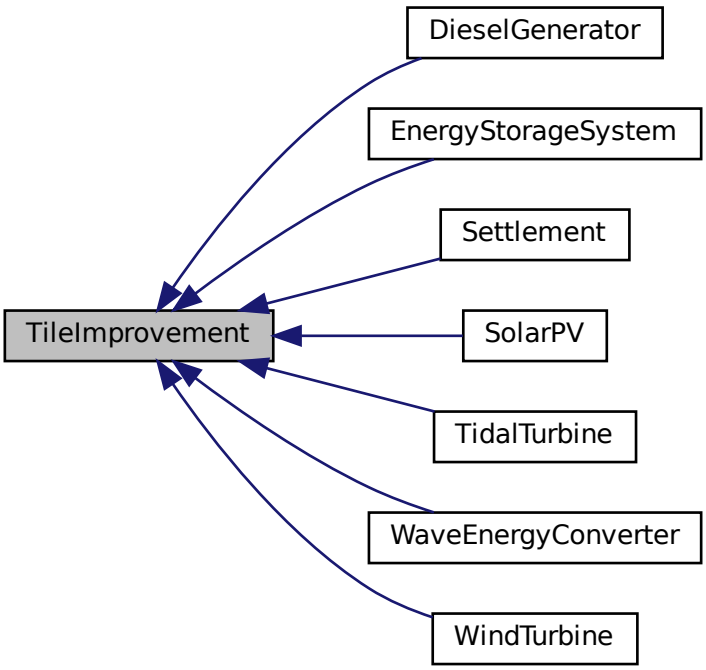
- header/TidalTurbine.h
- source/TidalTurbine.cpp

4.13 TileImprovement Class Reference

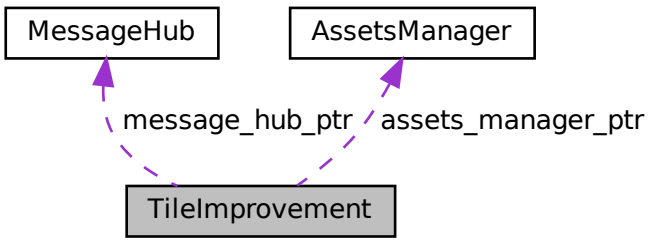
A base class for the tile improvement hierarchy.

`#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.
- sf::CircleShape dispatch_backing

  A backing circle for dispatch text illustration.
- sf::Text dispatch_text

  Text for illustrating dispatch.

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 __setUpDispatchIllustration (void)

  Helper method to set up and position dispatch 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.
- virtual void __repair (void)

  Helper method to repair a tile improvement.
- 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.
- void __drawDispatch (void)

  Helper method to draw dispatch illustration.

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.

```
726 {
727     //  1. set attributes
728 
729     //  1.1. protected
730     this->event_ptr = event_ptr;
731     this->render_window_ptr = render_window_ptr;
732 
733     this->assets_manager_ptr = assets_manager_ptr;
734     this->message_hub_ptr = message_hub_ptr;
735 
736     //  1.2. public
737     this->is_selected = true;
738     this->just_built = true;
739     this->production_menu_open = false;
740     this->upgrade_menu_open = false;
741     this->is_broken = false;
742 
743     this->just_upgraded = false;
744     this->upgrade_frame = 0;
745 
746     this->frame = 0;
747     this->credits = 0;
748     this->month = 1;
749     this->demand_MWh = 0;
750 
751     this->demand_vec_MWh.resize(30, 0);
752 
```

```
753     this->operation_maintenance_cost = 0;
754
755     this->tile_resource = tile_resource;
756
757     switch (this->tile_resource) {
758         case (0): {
759             this->tile_resource_scalar = 0.85;
760
761             break;
762         }
763
764
765         case (1): {
766             this->tile_resource_scalar = 0.925;
767
768             break;
769         }
770
771
772         case (2): {
773             this->tile_resource_scalar = 1;
774
775             break;
776         }
777
778
779         case (3): {
780             this->tile_resource_scalar = 1.075;
781
782             break;
783         }
784
785
786         case (4): {
787             this->tile_resource_scalar = 1.15;
788
789             break;
790         }
791
792
793         default: {
794             this->tile_resource_scalar = 1;
795         }
796     }
797
798     this->position_x = position_x;
799     this->position_y = position_y;
800
801     this->game_phase = "build settlement";
802
803     this->__setUpProductionMenu();
804     this->__setUpUpgradeMenu();
805     this->__setUpDispatchIllustration();
806
807     std::cout « "TileImprovement constructed at " « this « std::endl;
808
809     return;
810 }   /* TileImprovement() */
```

4.13.2.2 ~TileImprovement()

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

Destructor for the TileImprovement class.

```
1043 {
1044     std::cout « "TileImprovement at " « this « " destroyed" « std::endl;
1045
1046     return;
1047 }   /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __breakdown()

```
void TileImprovement::__breakdown (
            void  )  [protected]
```

Helper method to trigger an equipment breakdown.

```
431 {
432     this->is_broken = true;
433     this->is_running = false;
434     this->assets_manager_ptr->getSound("breakdown")->play();
435 
436     return;
437 }   /* __breakdown() */
```

4.13.3.2 __closeProductionMenu()

```
void TileImprovement::__closeProductionMenu (
            void  )  [protected]
```

Helper method to close the production menu.

```
407 {
408     if (not this->production_menu_open) {
409         return;
410     }
411 
412     this->production_menu_open = false;
413     this->assets_manager_ptr->getSound("build menu close")->play();
414 
415     return;
416 }   /* __closeProductionMenu() */
```

4.13.3.3 __closeUpgradeMenu()

```
void TileImprovement::__closeUpgradeMenu (
            void  )  [protected]
```

Helper method to close the build menu.

```
516 {
517     if (not this->upgrade_menu_open) {
518         return;
519     }
520 
521     this->upgrade_menu_open = false;
522     this->assets_manager_ptr->getSound("build menu close")->play();
523 
524     return;
525 }   /* __closeUpgradeMenu() */
```

4.13.3.4 __drawDispatch()

```
void TileImprovement::__drawDispatch (
            void  )  [protected]
```

Helper method to draw dispatch illustration.

```
647 {
648     double alpha = 255 * pow(cos((0.5 * M_PI * this->frame) / FRAMES_PER_SECOND), 2);
649 
650 
651     //  1. dispatch backing
652     sf::Color backing_colour = this->dispatch_backing.getFillColor();
653     backing_colour.a = alpha;
654 
655     this->dispatch_backing.setFillColor(backing_colour);
656     this->dispatch_backing.setOutlineColor(sf::Color(0, 0, 0, alpha));
657 
658     this->render_window_ptr->draw(this->dispatch_backing);
659 
660 
661     //  2. dispatch text
662     this->dispatch_text.setOrigin(
663         this->dispatch_text.getLocalBounds().width / 2,
664         this->dispatch_text.getLocalBounds().height / 2
665     );
666 
667     sf::Color text_colour = this->dispatch_text.getFillColor();
668     text_colour.a = alpha;
669 
670     this->dispatch_text.setFillColor(text_colour);
671 
672     this->render_window_ptr->draw(this->dispatch_text);
673 
674     return;
675 }   /* __drawDispatch() */
```

4.13.3.5 __handleKeyPressEvents()

```
void TileImprovement::__handleKeyPressEvents (
            void  )  [protected]
```

Helper method to handle key press events.

```
277 {
278     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
279         return;
280     }
281 
282     if (this->just_built) {
283         return;
284     }
285 
286     switch (this->event_ptr->key.code) {
287         case (sf::Keyboard::E): {
288             if (this->is_broken) {
289                 this->assets_manager_ptr->getSound("breakdown")->play();
290             }
291 
292             else {
293                 this->__openProductionMenu();
294             }
295 
296             break;
297         }
298 
299 
300         case (sf::Keyboard::R): {
301             if (this->is_broken) {
302                 this->__repair();
303             }
304 
305             break;
306         }
307 
308 
309         default: {
```

```
310             // do nothing!
311
312             break;
313         }
314     }
315
316     return;
317 }   /* __handleKeyPressEvents() */
```

4.13.3.6 __handleMouseButtonEvents()

```
void TileImprovement::__handleMouseButtonEvents (
            void  )  [protected]
```

Helper method to handle mouse button events.

```
332 {
333     if (this->tile_improvement_type == TileImprovementType :: SETTLEMENT) {
334         return;
335     }
336
337     if (this->just_built) {
338         return;
339     }
340
341     switch (this->event_ptr->mouseButton.button) {
342         case (sf::Mouse::Left): {
343             //...
344
345             break;
346         }
347
348
349         case (sf::Mouse::Right): {
350             //...
351
352             break;
353         }
354
355
356         default: {
357             // do nothing!
358
359             break;
360         }
361     }
362
363     return;
364 }   /* __handleMouseButtonEvents() */
```

4.13.3.7 __openProductionMenu()

```
void TileImprovement::__openProductionMenu (
            void  )  [protected]
```

Helper method to open the production menu.

```
379 {
380     if (this->production_menu_open) {
381         return;
382     }
383
384     if (this->upgrade_menu_open) {
385         this->__closeUpgradeMenu();
386     }
387
388     this->production_menu_open = true;
389     this->assets_manager_ptr->getSound("build menu open")->play();
390
391     return;
392 }   /* __openProductionMenu() */
```

4.13.3.8 __openUpgradeMenu()

```
void TileImprovement::__openUpgradeMenu (
            void  )  [protected]
```

Helper method to open the upgrade menu.

```
488 {
489     if (this->upgrade_menu_open) {
490         return;
491     }
492
493     if (this->production_menu_open) {
494         this->__closeProductionMenu();
495     }
496
497     this->upgrade_menu_open = true;
498     this->assets_manager_ptr->getSound("build menu open")->play();
499
500     return;
501 }   /* __openUpgradeMenu() */
```

4.13.3.9 __repair()

```
void TileImprovement::__repair (
            void  )  [protected], [virtual]
```

Helper method to repair a tile improvement.

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, and DieselGenerator.

```
452 {
453     this->health = 100;
454     if (this->is_broken) {
455         this->is_broken = false;
456         this->assets_manager_ptr->getSound("positive notification")->play();
457     }
458
459
460     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
461         this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
462     }
463
464     else {
465         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
466             this->tile_improvement_sprite_animated[i].setColor(
467                 sf::Color(255, 255, 255, 255)
468             );
469         }
470     }
471
472     return;
473 }   /* __repair() */
```

4.13.3.10 __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.

```
593 {
594     Message credits_spent_message;
595 
596     credits_spent_message.channel = GAME_CHANNEL;
597     credits_spent_message.subject = "credits spent";
598 
599     credits_spent_message.int_payload["credits spent"] = credits_spent;
600 
601     this->message_hub_ptr->sendMessage(credits_spent_message);
602 
603     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
604         << std::endl;
605     return;
606 }   /* __sendCreditsSpentMessage() */
```

4.13.3.11 __sendGameStateRequest()

```
void TileImprovement::__sendGameStateRequest (
            void  )  [protected]
```

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

```
566 {
567     Message game_state_request;
568 
569     game_state_request.channel = GAME_CHANNEL;
570     game_state_request.subject = "state request";
571 
572     this->message_hub_ptr->sendMessage(game_state_request);
573 
574     std::cout << "Game state request message sent by " << this << std::endl;
575     return;
576 }   /* __sendGameStateRequest() */
```

4.13.3.12 __sendInsufficientCreditsMessage()

```
void TileImprovement::__sendInsufficientCreditsMessage (
            void  )  [protected]
```

Helper method to format and send an insufficient credits message.

```
621 {
622     Message insufficient_credits_message;
623 
624     insufficient_credits_message.channel = GAME_CHANNEL;
625     insufficient_credits_message.subject = "insufficient credits";
626 
627     this->message_hub_ptr->sendMessage(insufficient_credits_message);
628 
629     std::cout << "Insufficient credits message sent by " << this << std::endl;
630 
631     return;
632 }   /* __sendInsufficientCreditsMessage() */
```

4.13.3.13 __sendTileStateRequest()

```
void TileImprovement::__sendTileStateRequest (
            void  )  [protected]
```

Helper method to format and send a request for the parent HexTile to send a tile state message.

```
541 {
542     Message tile_state_request;
543 
544     tile_state_request.channel = TILE_STATE_CHANNEL;
545     tile_state_request.subject = "state request";
546 
547     this->message_hub_ptr->sendMessage(tile_state_request);
548 
549     std::cout << "Tile state request sent by " << this << std::endl;
550     return;
551 }   /* __sendTileStateRequest() */
```

4.13.3.14 __setUpDispatchIllustration()

```
void TileImprovement::__setUpDispatchIllustration (
            void  )  [protected]
```

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

```
178 {
179     //  1. set up backing
180     this->dispatch_backing.setRadius(16);
181 
182     this->dispatch_backing.setOrigin(
183         this->dispatch_backing.getLocalBounds().width / 2,
184         this->dispatch_backing.getLocalBounds().height / 2
185     );
186 
187     this->dispatch_backing.setPosition(
188         this->position_x,
189         this->position_y
190     );
191 
192     this->dispatch_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
193     this->dispatch_backing.setOutlineThickness(2);
194     this->dispatch_backing.setOutlineColor(sf::Color(0, 0, 0, 255));
195 
196 
197     //  2. set up text
198     this->dispatch_text.setFont(*(assets_manager_ptr->getFont("Glass_TTY_VT220")));
199     this->dispatch_text.setFillColor(MONOCHROME_TEXT_GREEN);
200     this->dispatch_text.setCharacterSize(16);
201     this->dispatch_text.setPosition(
202         this->position_x,
203         this->position_y - 4
204     );
205 
206     return;
207 }   /* __setUpDispatchIllustration() */
```

4.13.3.15 __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() */
```

4.13.3.16 __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() */
```

4.13.3.17 __upgradeStorageCapacity()

```
void TileImprovement::__upgradeStorageCapacity (
            void  )  [protected]
```

Helper method to upgrade storage capacity.

```
222 {
223     if (this->credits < ENERGY_STORAGE_SYSTEM_BUILD_COST) {
```

```
224         std::cout « "Cannot add energy storage: insufficient credits (need "
225             « ENERGY_STORAGE_SYSTEM_BUILD_COST « " K)" « std::endl;
226 
227         this->__sendInsufficientCreditsMessage();
228         return;
229     }
230 
231     if (this->storage_level >= MAX_STORAGE_LEVELS) {
232         return;
233     }
234 
235     this->storage_level++;
236     this->storage_kWh += 200;
237 
238     this->storage_upgrade_sprite_vec.push_back(
239         sf::Sprite(
240             *(this->assets_manager_ptr->getTexture("storage level"))
241         )
242     );
243 
244     this->storage_upgrade_sprite_vec.back().setOrigin(
245         this->storage_upgrade_sprite_vec.back().getLocalBounds().width / 2,
246         this->storage_upgrade_sprite_vec.back().getLocalBounds().height
247     );
248 
249     this->storage_upgrade_sprite_vec.back().setPosition(
250         this->position_x + 18,
251         this->position_y + 25 - 7 * this->storage_upgrade_sprite_vec.size()
252     );
253 
254     this->just_upgraded = true;
255 
256     this->assets_manager_ptr->getSound("upgrade")->play();
257 
258     this->__sendCreditsSpentMessage(ENERGY_STORAGE_SYSTEM_BUILD_COST);
259     this->__sendTileStateRequest();
260 
261     return;
262 }   /* __upgradeStorageCapacity() */
```

4.13.3.18 advanceTurn()

```
virtual void TileImprovement::advanceTurn (
             void  )  [inline], [virtual]
```

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, and DieselGenerator.

```
191 {return;}
```

4.13.3.19 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.

```
914 {
915     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
916         int alpha = this->tile_improvement_sprite_static.getColor().a;
917 
918         alpha += 0.08 * FRAMES_PER_SECOND;
919 
920         this->tile_improvement_sprite_static.setColor(
921             sf::Color(255, 255, 255, alpha)
922         );
923 
924         this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);
```

```
925
926         if (
927             (alpha >= 255) or
928             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
929         ) {
930             this->tile_improvement_sprite_static.setColor(
931                 sf::Color(255, 255, 255, 255)
932             );
933
934             this->tile_improvement_sprite_static.setPosition(
935                 this->position_x,
936                 this->position_y + 12
937             );
938
939             this->just_built = false;
940             this->assets_manager_ptr->getSound("place improvement")->play();
941         }
942
943         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
944     }
945
946
947     else {
948         int alpha = 0;
949
950         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
951             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
952
953             alpha += 0.08 * FRAMES_PER_SECOND;
954
955             this->tile_improvement_sprite_animated[i].setColor(
956                 sf::Color(255, 255, 255, alpha)
957             );
958
959             this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
960
961             if (
962                 (alpha >= 255) or
963                 (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
964             ) {
965                 this->tile_improvement_sprite_animated[i].setColor(
966                     sf::Color(255, 255, 255, 255)
967                 );
968
969                 this->tile_improvement_sprite_animated[i].setPosition(
970                     this->position_x,
971                     this->position_y + 12
972                 );
973             }
974
975             this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
976         }
977
978         if (
979             (alpha >= 255) or
980             (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
981         ) {
982             this->just_built = false;
983             this->assets_manager_ptr->getSound("place improvement")->play();
984
985             switch (this->tile_improvement_type) {
986                 case (TileImprovementType :: WIND_TURBINE): {
987                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
988                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
989                         this->tile_improvement_sprite_animated[i].move(0, -32);
990                     }
991
992                     break;
993                 }
994
995
996                 case (TileImprovementType :: TIDAL_TURBINE): {
997                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
998                         this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
999                         this->tile_improvement_sprite_animated[i].move(0, -19);
1000                     }
1001
1002                     break;
1003                 }
1004
1005
1006                 case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
1007                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1008                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
1009                         this->tile_improvement_sprite_animated[i].move(0, -32);
1010                     }
1011
```

```
1012                     break;
1013                 }
1014
1015
1016                 default: {
1017                     // do nothing!
1018
1019                     break;
1020                 }
1021             }
1022         }
1023     }
1024
1025
1026     this->frame++;
1027     return;
1028 }   /* draw() */
```

4.13.3.20 getTileOptionsSubstring()

```
virtual std::string TileImprovement::getTileOptionsSubstring (
            void  )  [inline], [virtual]
```

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, SolarPV, Settlement, EnergyStorageSystem, and DieselGenerator.

```
195 {return "";}
```

4.13.3.21 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.

```
854 {
855     if (this->event_ptr->type == sf::Event::KeyPressed) {
856         this->__handleKeyPressEvents();
857     }
858
859     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
860         this->__handleMouseButtonEvents();
861     }
862
863     return;
864 }   /* processEvent() */
```

4.13.3.22 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.

```
879 {
880     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
881         Message game_state_message = this->message_hub_ptr->receiveMessage(
882             GAME_STATE_CHANNEL
883         );
884 
885         if (game_state_message.subject == "turn advance") {
886             this->credits = game_state_message.int_payload["credits"];
887             this->month = game_state_message.int_payload["month"];
888             this->demand_MWh = game_state_message.int_payload["demand_MWh"];
889 
890             this->advanceTurn();
891 
892             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
893             std::cout << "Turn advance message read and passed by " << this << std::endl;
894         }
895     }
896 
897     return;
898 }   /* processMessage() */
```

4.13.3.23 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.

```
827 {
828     this->is_selected = is_selected;
829 
830     if ((not is_selected) and this->production_menu_open) {
831         this->__closeProductionMenu();
832     }
833 
834     if ((not is_selected) and this->upgrade_menu_open) {
835         this->__closeUpgradeMenu();
836     }
837 
838     return;
839 }   /* setIsSelected() */
```

4.13.3.24 update()

```
virtual void TileImprovement::update (
            void  )  [inline], [virtual]
```

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, and SolarPV.
193 {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 dispatch_backing

```
sf::CircleShape TileImprovement::dispatch_backing
```

A backing circle for dispatch text illustration.

4.13.4.6 dispatch_text

```
sf::Text TileImprovement::dispatch_text
```

Text for illustrating dispatch.

4.13.4.7 event_ptr

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

A pointer to the event class.

4.13.4.8 frame

```
unsigned long long int TileImprovement::frame
```

The current frame of this object.

4.13.4.9 game_phase

```
std::string TileImprovement::game_phase
```

The current phase of the game.

4.13.4.10 health

```
int TileImprovement::health
```

The health of the improvement.

4.13.4.11 is_broken

```
bool TileImprovement::is_broken
```

A boolean which indicated whether or not improvement is broken.

4.13.4.12 is_running

```
bool TileImprovement::is_running
```

A boolean which indicates whether or not the improvement is running.

4.13.4.13 is_selected

```
bool TileImprovement::is_selected
```

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

4.13.4.14 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.15 just_upgraded

```
bool TileImprovement::just_upgraded
```

A boolean which indicates that the improvement was just upgraded.

4.13.4.16 message_hub_ptr

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

A pointer to the message hub.

4.13.4.17 month

```
int TileImprovement::month
```

The current month of play.

4.13.4.18 operation_maintenance_cost

```
int TileImprovement::operation_maintenance_cost
```

The operation and maintenance costs for this turn.

4.13.4.19 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.20 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.21 production_menu_backing

```
sf::RectangleShape TileImprovement::production_menu_backing
```

A backing for the production menu.

4.13.4.22 production_menu_backing_text

```
sf::Text TileImprovement::production_menu_backing_text
```

Text for the production menu backing.

4.13.4.23 production_menu_open

```
bool TileImprovement::production_menu_open
```

A boolean which indicates whether or not the production menu is open.

4.13.4.24 render_window_ptr

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

A pointer to the render window.

4.13.4.25 storage_kWh

```
int TileImprovement::storage_kWh
```

The rated energy capacity [kWh] of the storage.

4.13.4.26 storage_level

```
int TileImprovement::storage_level
```

The level of storage installed alongside the tile improvement.

4.13.4.27 storage_upgrade_sprite

```
sf::Sprite TileImprovement::storage_upgrade_sprite
```

A sprite for illustrating storage (in upgrade menu).

4.13.4.28 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).

4.13.4.29 tile_improvement_sprite_animated

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

An animated sprite, for the ContextMenu visual screen.

4.13.4.30 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.31 tile_improvement_string

```
std::string TileImprovement::tile_improvement_string
```

A string representation of the tile improvement type.

4.13.4.32 tile_improvement_type

```
TileImprovementType TileImprovement::tile_improvement_type
```

The type of the tile improvement.

4.13.4.33 tile_resource

```
int TileImprovement::tile_resource
```

The renewable resource quality of the tile.

4.13.4.34 tile_resource_scalar

```
double TileImprovement::tile_resource_scalar
```

A scalar associated with the renewable resource quality.

4.13.4.35 upgrade_arrow_sprite

```
sf::Sprite TileImprovement::upgrade_arrow_sprite
```

An upgrade arrow sprite.

4.13.4.36 upgrade_frame

```
int TileImprovement::upgrade_frame
```

The frame of the upgrade animation.

4.13.4.37 upgrade_level

```
int TileImprovement::upgrade_level
```

The upgrade level of the improvement.

4.13.4.38 upgrade_menu_backing

```
sf::RectangleShape TileImprovement::upgrade_menu_backing
```

A backing for the upgrade menu.

4.13.4.39 upgrade_menu_backing_text

```
sf::Text TileImprovement::upgrade_menu_backing_text
```

Text for the upgrade menu backing.

4.13.4.40 upgrade_menu_open

```
bool TileImprovement::upgrade_menu_open
```

A boolean which indicates whether or not the build menu is open.

4.13.4.41 upgrade_plus_sprite

```
sf::Sprite TileImprovement::upgrade_plus_sprite
```

An upgrade plus sprite.

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

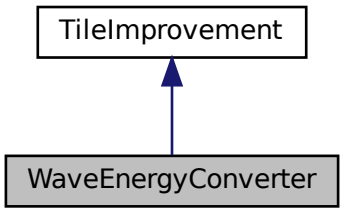
- header/TileImprovement.h
- source/TileImprovement.cpp

4.14 WaveEnergyConverter Class Reference

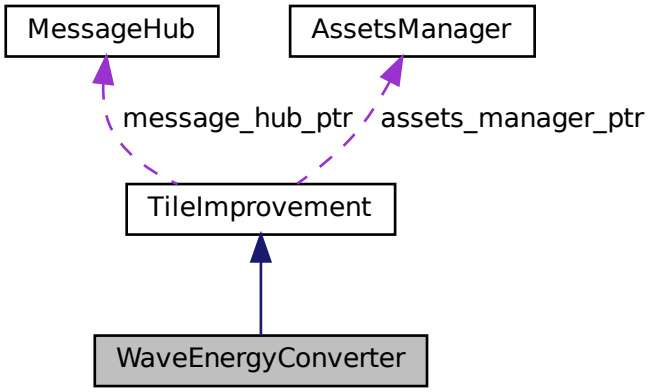
A settlement class (child class of TileImprovement).

`#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.
- double bobbing_y

  The bobbing extent of the wave energy converter.
- 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 __repair (void)

  Helper method to repair the wave energy converter.
- 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.

```
763   :
764 TileImprovement(
765     position_x,
766     position_y,
767     tile_resource,
768     event_ptr,
769     render_window_ptr,
770     assets_manager_ptr,
771     message_hub_ptr
772 )
773 {
774     //  1. set attributes
775
776     //  1.1. private
777     //...
778
779     //  1.2. public
780     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
781
782     this->is_running = false;
783
784     this->health = 100;
785
786     this->capacity_kW = 100;
787     this->upgrade_level = 1;
788
789     this->storage_kWh = 0;
790     this->storage_level = 0;
791
792     this->production_MWh = 0;
793     this->dispatch_MWh = 0;
794     this->dispatchable_MWh = 0;
795
796     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
797
798     this->bobbing_y = 4;
799
800     this->capacity_factor_vec.resize(30, 0);
801     this->production_vec_MWh.resize(30, 0);
802     this->dispatch_vec_MWh.resize(30, 0);
803
804     this->tile_improvement_string = "WAVE ENERGY";
805
806     this->__setUpTileImprovementSpriteAnimated();
807     this->__computeCapacityFactors();
808     this->update();
809
810     std::cout « "WaveEnergyConverter constructed at " « this « std::endl;
811
812     return;
813 }   /* WaveEnergyConverter() */
```

4.14.2.2 ~WaveEnergyConverter()

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

Destructor for the WaveEnergyConverter class.

```
1182 {
1183     std::cout « "WaveEnergyConverter at " « this « " destroyed" « std::endl;
1184
1185     return;
1186 }   /* ~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.

```
307 {
308     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
309     std::default_random_engine generator(seed);
310 
311     double mean =
312         this->tile_resource_scalar * MEAN_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
313 
314     double stdev = STDEV_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
315 
316     if (this->tile_resource_scalar > 1) {
317         stdev /= this->tile_resource_scalar;
318     }
319 
320     std::normal_distribution<double> normal_dist(mean, stdev);
321 
322     double capacity_factor = 0;
323 
324     for (int i = 0; i < 30; i++) {
325         capacity_factor = normal_dist(generator);
326 
327         if (capacity_factor < 0) {
328             capacity_factor = 0;
329         }
330 
331         this->capacity_factor_vec[i] = capacity_factor;
332     }
333 
334     return;
335 }   /* __computeCapacityFactors() */
```

4.14.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
378 {
379     double stored_energy_MWh = 0;
380     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
381 
382     double demand_MWh = 0;
383     double production_MWh = 0;
384     double dispatchable_MWh = 0;
```

```
385     double difference_MWh = 0;
386
387     double room_MWh = 0;
388
389     for (int i = 0; i < 30; i++) {
390         demand_MWh = this->demand_vec_MWh[i];
391         production_MWh = this->production_vec_MWh[i];
392
393         if (production_MWh <= demand_MWh) {
394             this->dispatch_vec_MWh[i] = production_MWh;
395             dispatchable_MWh += this->dispatch_vec_MWh[i];
396
397             difference_MWh = demand_MWh - production_MWh;
398
399             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
400                 if (difference_MWh > stored_energy_MWh) {
401                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
402                     dispatchable_MWh += stored_energy_MWh;
403                     stored_energy_MWh = 0;
404                 }
405
406                 else {
407                     this->dispatch_vec_MWh[i] += difference_MWh;
408                     dispatchable_MWh += difference_MWh;
409                     stored_energy_MWh -= difference_MWh;
410                 }
411             }
412         }
413
414         else {
415             this->dispatch_vec_MWh[i] = demand_MWh;
416             dispatchable_MWh += this->dispatch_vec_MWh[i];
417
418             difference_MWh = production_MWh - demand_MWh;
419
420             if (
421                 (storage_capacity_MWh > 0) and
422                 (stored_energy_MWh < storage_capacity_MWh)
423             ) {
424                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
425
426                 if (difference_MWh > room_MWh) {
427                     stored_energy_MWh += room_MWh;
428                 }
429
430                 else {
431                     stored_energy_MWh += difference_MWh;
432                 }
433             }
434         }
435     }
436
437     this->dispatchable_MWh = round(dispatchable_MWh);
438
439     if (this->dispatch_MWh != this->dispatchable_MWh) {
440         this->dispatch_MWh = this->dispatchable_MWh;
441     }
442
443     return;
444 }   /* __computeDispatch() */
```

4.14.3.4 __computeProduction()

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

Helper method to compute production values.

```
350 {
351     double production_MWh = 0;
352
353     for (int i = 0; i < 30; i++) {
354         this->production_vec_MWh[i] =
355             this->max_daily_production_MWh * this->capacity_factor_vec[i];
356
357         production_MWh += this->production_vec_MWh[i];
358     }
359
360     this->production_MWh = round(production_MWh);
```

```
361
362     return;
363 }   /* __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.

```
584 {
585     //  1. draw power capacity upgrade sprite
586     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
587         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
588         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
589
590         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
591         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
592
593         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
594         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
595
596         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
597
598         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
599         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
600         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
601     }
602
603     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
604
605
606     //  2. draw power capacity upgrade text
607     //                  16 char line = "               \n"
608     std::string power_upgrade_string = "POWER CAPACITY  \n";
609     power_upgrade_string            += "               \n";
610
611     power_upgrade_string            += "CAPACITY:  ";
612     power_upgrade_string            += std::to_string(this->capacity_kW);
613     power_upgrade_string            += " kW\n";
614
615     power_upgrade_string            += "LEVEL:     ";
616     power_upgrade_string            += std::to_string(this->upgrade_level);
617     power_upgrade_string            += "\n\n";
618
619     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
620         power_upgrade_string        += "[W]: + 100 kW (";
621         power_upgrade_string        +=  std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
622         power_upgrade_string        += " K)\n";
623     }
624
625     else {
626         power_upgrade_string        += " * MAX LEVEL *  \n";
627     }
628
629     sf::Text power_upgrade_text = sf::Text(
630         power_upgrade_string,
631         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
632         16
633     );
634
635     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
636     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
637     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
638
639     this->render_window_ptr->draw(power_upgrade_text);
640
641
642     //  3. draw energy capacity (storage) upgrade sprite
643     this->render_window_ptr->draw(this->storage_upgrade_sprite);
644     this->render_window_ptr->draw(this->upgrade_plus_sprite);
645
646
```

```
647     //  4. draw energy capacity (storage) upgrade text
648     //                   16 char line = "                \n"
649     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
650     energy_upgrade_string            += "                \n";
651 
652     energy_upgrade_string            += "CAPACITY:  ";
653     energy_upgrade_string            += std::to_string(this->storage_level * 200);
654     energy_upgrade_string            += " kWh\n";
655 
656     energy_upgrade_string            += "LEVEL:     ";
657     energy_upgrade_string            += std::to_string(this->storage_level);
658     energy_upgrade_string            += "\n\n";
659 
660     if (this->storage_level < MAX_STORAGE_LEVELS) {
661         energy_upgrade_string        += "[D]: + 200 kWh (";
662         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
663         energy_upgrade_string        += " K)\n";
664     }
665 
666     else {
667         energy_upgrade_string += " * MAX LEVEL *  \n";
668     }
669 
670     sf::Text energy_upgrade_text = sf::Text(
671         energy_upgrade_string,
672         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
673         16
674     );
675 
676     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
677     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
678     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
679 
680     this->render_window_ptr->draw(energy_upgrade_text);
681 
682     return;
683 }   /* __drawUpgradeOptions() */
```

4.14.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
459 {
460     if (this->just_built) {
461         return;
462     }
463 
464     switch (this->event_ptr->key.code) {
465         case (sf::Keyboard::U): {
466             this->__openUpgradeMenu();
467 
468             break;
469         }
470 
471 
472         case (sf::Keyboard::W): {
473             if (this->production_menu_open) {
474                 this->dispatch_MWh++;
475 
476                 if (this->dispatch_MWh > this->dispatchable_MWh) {
477                     this->dispatch_MWh = 0;
478                 }
479 
480                 this->__computeProductionCosts();
481                 this->assets_manager_ptr->getSound("interface click")->play();
482             }
483 
484             else if (this->upgrade_menu_open) {
485                 this->__upgradePowerCapacity();
486             }
487 
488             break;
489         }
490 
491 
492         case (sf::Keyboard::S): {
```

```
493             if (this->production_menu_open) {
494                 this->dispatch_MWh--;
495 
496                 if (this->dispatch_MWh < 0) {
497                     this->dispatch_MWh = this->dispatchable_MWh;
498                 }
499 
500                 this->__computeProductionCosts();
501                 this->assets_manager_ptr->getSound("interface click")->play();
502             }
503 
504             break;
505         }
506 
507 
508         case (sf::Keyboard::D): {
509             if (this->upgrade_menu_open) {
510                 this->__upgradeStorageCapacity();
511                 this->__computeProduction();
512                 this->__computeDispatch();
513             }
514 
515             break;
516         }
517 
518 
519         default: {
520             // do nothing!
521 
522             break;
523         }
524     }
525 
526     return;
527 }   /* __handleKeyPressEvents() */
```

4.14.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
542 {
543     if (this->just_built) {
544         return;
545     }
546     switch (this->event_ptr->mouseButton.button) {
547         case (sf::Mouse::Left): {
548             //...
549 
550             break;
551         }
552 
553 
554         case (sf::Mouse::Right): {
555             //...
556 
557             break;
558         }
559 
560 
561         default: {
562             // do nothing!
563 
564             break;
565         }
566     }
567 
568     return;
569 }   /* __handleMouseButtonEvents() */
```

4.14.3.10 __repair()

```
void WaveEnergyConverter::__repair (
            void  )  [private], [virtual]
```

Helper method to repair the wave energy converter.

Reimplemented from TileImprovement.

```
274 {
275     if (this->credits < WAVE_ENERGY_CONVERTER_BUILD_COST) {
276         std::cout « "Cannot repair wave energy converter: insufficient credits (need "
277             « WAVE_ENERGY_CONVERTER_BUILD_COST « " K)" « std::endl;
278 
279         this->__sendInsufficientCreditsMessage();
280         return;
281     }
282 
283     TileImprovement :: __repair();
284 
285     this->just_upgraded = true;
286 
287     this->__sendCreditsSpentMessage(WAVE_ENERGY_CONVERTER_BUILD_COST);
288     this->__sendTileStateRequest();
289     this->__sendGameStateRequest();
290 
291     return;
292 }   /* __repair() */
```

4.14.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
698 {
699     Message improvement_state_message;
700 
701     improvement_state_message.channel = GAME_CHANNEL;
702     improvement_state_message.subject = "improvement state";
703 
704     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
705     improvement_state_message.int_payload["operation_maintenance_cost"] =
706         this->operation_maintenance_cost;
707 
708     this->message_hub_ptr->sendMessage(improvement_state_message);
709 
710     std::cout « "Improvement state message sent by " « this « std::endl;
711 
712     return;
713 }   /* __sendImprovementStateMessage() */
```

4.14.3.12 __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() */
```

4.14.3.13 __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     TileImprovement :: __repair();
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() */
```

4.14.3.14 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
918 {
919     //  1. send improvement state message
920     this->__sendImprovementStateMessage();
921 
922     //  2. update
923     this->__computeCapacityFactors();
924     this->update();
925 
926     //  3. handle start/stop
927     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
928         this->is_running = true;
929     }
930 
931     else if (this->is_running and (this->dispatch_MWh <= 0)) {
932         this->is_running = false;
933     }
934 
935     //  4. handle equipment health
936     if (this->is_running) {
937         this->health--;
938 
939         if (this->health <= 0) {
940             this->__breakdown();
941         }
942     }
943 
944     //  5. send tile state request (if selected)
945     if (this->is_selected) {
946         this->__sendTileStateRequest();
947     }
948 
949     return;
950 }   /* advanceTurn() */
```

4.14.3.15 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.

```
1039 {
1040     //  1. if just built, call base method and return
1041     if (this->just_built) {
1042         TileImprovement :: draw();
1043 
1044         return;
1045     }
1046 
1047 
1048     //  2. handle upgrade effects
1049     if (this->just_upgraded) {
1050         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1051             this->tile_improvement_sprite_animated[i].setColor(
1052                 sf::Color(
1053                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1054                     255,
1055                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1056                     255
1057                 )
1058             );
1059 
1060             this->tile_improvement_sprite_animated[i].setScale(
1061                 sf::Vector2f(
1062                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1063                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1064                 )
1065             );
1066         }
1067 
1068         this->upgrade_frame++;
1069     }
```

```
1070
1071        if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1072            for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1073                this->tile_improvement_sprite_animated[i].setColor(
1074                    sf::Color(255,255,255,255)
1075                );
1076
1077                this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1078            }
1079
1080            this->just_upgraded = false;
1081            this->upgrade_frame = 0;
1082        }
1083
1084
1085        //  3. draw first element of animated sprite
1086        this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1087
1088
1089        //  4. draw second element of animated sprite
1090        if (this->is_running) {
1091            this->tile_improvement_sprite_animated[0].setPosition(
1092                this->position_x,
1093                this->position_y + this->bobbing_y * cos(
1094                    (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1095                )
1096            );
1097
1098            this->tile_improvement_sprite_animated[1].setPosition(
1099                this->position_x,
1100                this->position_y + 1.25 * this->bobbing_y * sin(
1101                    (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1102                )
1103            );
1104        }
1105
1106        else {
1107            for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1108                this->tile_improvement_sprite_animated[i].setPosition(
1109                    this->position_x,
1110                    this->position_y + this->bobbing_y * cos(
1111                        (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1112                    )
1113                );
1114            }
1115        }
1116
1117        this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1118
1119
1120        //  5. draw storage upgrades
1121        for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1122            this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1123        }
1124
1125
1126        //  6. handle dispatch illustration
1127        if (this->dispatch_MWh > 0) {
1128            this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1129            this->__drawDispatch();
1130        }
1131
1132
1133        //  7. draw production menu
1134        if (this->production_menu_open) {
1135            this->render_window_ptr->draw(this->production_menu_backing);
1136            this->render_window_ptr->draw(this->production_menu_backing_text);
1137
1138            this->__drawProductionMenu();
1139        }
1140
1141
1142        //  8. draw upgrade menu
1143        if (this->upgrade_menu_open) {
1144            this->render_window_ptr->draw(this->upgrade_menu_backing);
1145            this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1146
1147            this->__drawUpgradeOptions();
1148        }
1149
1150
1151        //  9. handle broken effects
1152        if (this->is_broken) {
1153            for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1154                this->tile_improvement_sprite_animated[i].setColor(
1155                    sf::Color(
1156                        255,
```

```
1157                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1158                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1159                     255
1160                 )
1161             );
1162         }
1163     }
1164
1165     this->frame++;
1166     return;
1167 }   /* draw() */
```

4.14.3.16 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
830 {
831     //                  32 char x 17 line console "--------------------------------\n";
832     std::string options_substring                = "CAPACITY:      ";
833     options_substring                           += std::to_string(this->capacity_kW);
834     options_substring                           += " kW (level ";
835     options_substring                           += std::to_string(this->upgrade_level);
836     options_substring                           += ")\n";
837
838     options_substring                           += "PRODUCTION:    ";
839     options_substring                           += std::to_string(this->production_MWh);
840     options_substring                           += " MWh\n";
841
842     options_substring                           += "DISPATCHABLE:  ";
843     options_substring                           += std::to_string(this->dispatchable_MWh);
844     options_substring                           += " MWh\n";
845
846     options_substring                           += "HEALTH:      ";
847     options_substring                           += std::to_string(this->health);
848     options_substring                           += "/100";
849
850     if (this->health <= 0) {
851         options_substring                       += " ** BROKEN! **\n";
852     }
853
854     else {
855         options_substring                       += "\n";
856     }
857
858     options_substring                           += "                                \n";
859     options_substring                           += " **** WAVE ENERGY OPTIONS ****  \n";
860     options_substring                           += "                                \n";
861
862     if (this->is_broken) {
863         options_substring                       += "     [R]:  REPAIR (";
864         options_substring                       += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
865         options_substring                       += " K)\n";
866     }
867
868     else {
869         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
870     }
871
872     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
873     options_substring                           += "HOLD [P]:  SCRAP (";
874     options_substring                           += std::to_string(SCRAP_COST);
875     options_substring                           += " K)";
876
877     return options_substring;
878 }   /* getTileOptionsSubstring() */
```

4.14.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
990 {
991     TileImprovement :: processEvent();
992
993     if (this->event_ptr->type == sf::Event::KeyPressed) {
994         this->__handleKeyPressEvents();
995     }
996
997     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
998         this->__handleMouseButtonEvents();
999     }
1000
1001     return;
1002 }   /* processEvent() */
```

4.14.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1017 {
1018     TileImprovement :: processMessage();
1019
1020     //...
1021
1022     return;
1023 }   /* processMessage() */
```

4.14.3.19 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.

```
895 {
896     TileImprovement :: setIsSelected(is_selected);
897
898     if (this->is_running and this->is_selected) {
899         this->assets_manager_ptr->getSound("ocean waves")->play();
900     }
901
```

```
902     return;
903 }   /* setIsSelected() */
```

4.14.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
965 {
966     this->__computeProduction();
967     this->__computeProductionCosts();
968     this->__computeDispatch();
969 
970     if (this->is_selected) {
971         this->__sendTileStateRequest();
972     }
973 
974     return;
975 }   /* update() */
```

4.14.4 Member Data Documentation

4.14.4.1 bobbing_y

```
double WaveEnergyConverter::bobbing_y
```

The bobbing extent of the wave energy converter.

4.14.4.2 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.14.4.3 capacity_kW

```
int WaveEnergyConverter::capacity_kW
```

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

4.14.4.4 dispatch_MWh

```
int WaveEnergyConverter::dispatch_MWh
```

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

4.14.4.5 dispatch_vec_MWh

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

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

4.14.4.6 dispatchable_MWh

```
int WaveEnergyConverter::dispatchable_MWh
```

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

4.14.4.7 max_daily_production_MWh

```
double WaveEnergyConverter::max_daily_production_MWh
```

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

4.14.4.8 production_MWh

```
int WaveEnergyConverter::production_MWh
```

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

4.14.4.9 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:

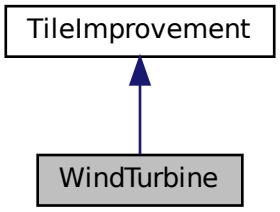
- header/WaveEnergyConverter.h
- source/WaveEnergyConverter.cpp

4.15 WindTurbine Class Reference

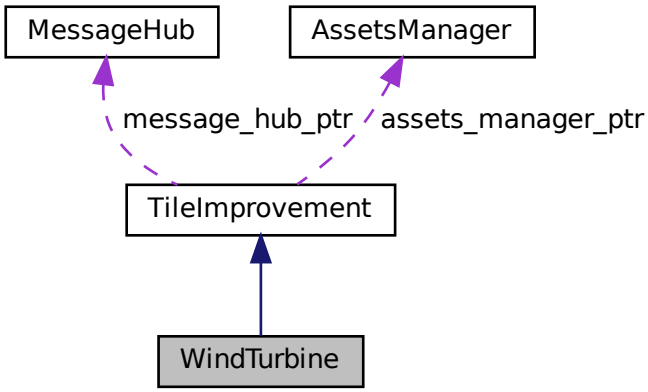
A settlement class (child class of TileImprovement).

`#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.
- 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 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 __repair (void)

  Helper method to repair the wind turbine.
- 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.

```
770   :
771 TileImprovement(
772     position_x,
773     position_y,
774     tile_resource,
775     event_ptr,
776     render_window_ptr,
777     assets_manager_ptr,
778     message_hub_ptr
779 )
780 {
781     //  1. set attributes
782
783     //  1.1. private
784     //...
785
786     //  1.2. public
787     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
788
789     this->is_running = false;
790
791     this->health = 100;
792
793     this->capacity_kW = 100;
794     this->upgrade_level = 1;
795
796     this->storage_kWh = 0;
797     this->storage_level = 0;
798
799     this->production_MWh = 0;
800     this->dispatch_MWh = 0;
801     this->dispatchable_MWh = 0;
802
803     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
804
805     this->rotor_drotation = 256 * SECONDS_PER_FRAME;
806
807     this->capacity_factor_vec.resize(30, 0);
808     this->production_vec_MWh.resize(30, 0);
809     this->dispatch_vec_MWh.resize(30, 0);
810
811     this->tile_improvement_string = "WIND TURBINE";
812
813     this->__setUpTileImprovementSpriteAnimated();
814     this->__computeCapacityFactors();
815     this->update();
816
817     std::cout « "WindTurbine constructed at " « this « std::endl;
818
819     return;
820 }   /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

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

Destructor for the WindTurbine class.

```
1168 {
1169     std::cout « "WindTurbine at " « this « " destroyed" « std::endl;
1170
1171     return;
1172 }   /* ~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.

```
307 {
308     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
309     std::default_random_engine generator(seed);
310 
311     double mean =
312         this->tile_resource_scalar * MEAN_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
313 
314     double stdev = STDEV_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
315 
316     if (this->tile_resource_scalar > 1) {
317         stdev /= this->tile_resource_scalar;
318     }
319 
320     std::normal_distribution<double> normal_dist(mean, stdev);
321 
322     double capacity_factor = 0;
323 
324     for (int i = 0; i < 30; i++) {
325         capacity_factor = normal_dist(generator);
326 
327         if (capacity_factor < 0) {
328             capacity_factor = 0;
329         }
330 
331         this->capacity_factor_vec[i] = capacity_factor;
332     }
333 
334     return;
335 }   /* __computeCapacityFactors() */
```

4.15.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
378 {
379     std::cout « "WindTurbine :: __computeDispatch()" « std::endl;
380 
381     double stored_energy_MWh = 0;
382     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
383 
384     double demand_MWh = 0;
```

```
385     double production_MWh = 0;
386     double dispatchable_MWh = 0;
387     double difference_MWh = 0;
388 
389     double room_MWh = 0;
390 
391     for (int i = 0; i < 30; i++) {
392         demand_MWh = this->demand_vec_MWh[i];
393         production_MWh = this->production_vec_MWh[i];
394 
395         if (production_MWh <= demand_MWh) {
396             this->dispatch_vec_MWh[i] = production_MWh;
397             dispatchable_MWh += this->dispatch_vec_MWh[i];
398 
399             difference_MWh = demand_MWh - production_MWh;
400 
401             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
402                 if (difference_MWh > stored_energy_MWh) {
403                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
404                     dispatchable_MWh += stored_energy_MWh;
405                     stored_energy_MWh = 0;
406                 }
407 
408                 else {
409                     this->dispatch_vec_MWh[i] += difference_MWh;
410                     dispatchable_MWh += difference_MWh;
411                     stored_energy_MWh -= difference_MWh;
412                 }
413             }
414         }
415 
416         else {
417             this->dispatch_vec_MWh[i] = demand_MWh;
418             dispatchable_MWh += this->dispatch_vec_MWh[i];
419 
420             difference_MWh = production_MWh - demand_MWh;
421 
422             if (
423                 (storage_capacity_MWh > 0) and
424                 (stored_energy_MWh < storage_capacity_MWh)
425             ) {
426                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
427 
428                 if (difference_MWh > room_MWh) {
429                     stored_energy_MWh += room_MWh;
430                 }
431 
432                 else {
433                     stored_energy_MWh += difference_MWh;
434                 }
435             }
436         }
437     }
438 
439     this->dispatchable_MWh = round(dispatchable_MWh);
440 
441     if (this->dispatch_MWh != this->dispatchable_MWh) {
442         this->dispatch_MWh = this->dispatchable_MWh;
443     }
444 
445     return;
446 }   /* __computeDispatch() */
```

4.15.3.4 __computeProduction()

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

Helper method to compute production values.

```
350 {
351     double production_MWh = 0;
352 
353     for (int i = 0; i < 30; i++) {
354         this->production_vec_MWh[i] =
355             this->max_daily_production_MWh * this->capacity_factor_vec[i];
356 
357         production_MWh += this->production_vec_MWh[i];
358     }
```

```
359
360     this->production_MWh = round(production_MWh);
361
362     return;
363 }   /* __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.

```
587 {
588     //  1. draw power capacity upgrade sprite
589     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
590         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
591         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
592
593         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
594         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
595
596         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
597         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
598
599         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
600         this->tile_improvement_sprite_animated[i].setRotation(0);
601
602         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
603
604         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
605         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
606         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
607         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
608     }
609
610     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
611
612
613     //  2. draw power capacity upgrade text
614     //                  16 char line = "                \n"
615     std::string power_upgrade_string = "POWER CAPACITY  \n";
616     power_upgrade_string            += "                \n";
617
618     power_upgrade_string            += "CAPACITY:  ";
619     power_upgrade_string            += std::to_string(this->capacity_kW);
620     power_upgrade_string            += " kW\n";
621
622     power_upgrade_string            += "LEVEL:     ";
623     power_upgrade_string            += std::to_string(this->upgrade_level);
624     power_upgrade_string            += "\n\n";
625
626     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
627         power_upgrade_string        += "[W]: + 100 kW (";
628         power_upgrade_string        +=  std::to_string(WIND_TURBINE_BUILD_COST);
629         power_upgrade_string        += " K)\n";
630     }
631
632     else {
633         power_upgrade_string        += " * MAX LEVEL *  \n";
634     }
635
636     sf::Text power_upgrade_text = sf::Text(
637         power_upgrade_string,
638         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
639         16
640     );
641
642     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
643     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
644     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
645
646     this->render_window_ptr->draw(power_upgrade_text);
647
```

```
648
649     //  3. draw energy capacity (storage) upgrade sprite
650     this->render_window_ptr->draw(this->storage_upgrade_sprite);
651     this->render_window_ptr->draw(this->upgrade_plus_sprite);
652
653
654     //  4. draw energy capacity (storage) upgrade text
655     //                   16 char line = "                \n"
656     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
657     energy_upgrade_string            += "                \n";
658
659     energy_upgrade_string            += "CAPACITY:  ";
660     energy_upgrade_string            += std::to_string(this->storage_level * 200);
661     energy_upgrade_string            += " kWh\n";
662
663     energy_upgrade_string            += "LEVEL:     ";
664     energy_upgrade_string            += std::to_string(this->storage_level);
665     energy_upgrade_string            += "\n\n";
666
667     if (this->storage_level < MAX_STORAGE_LEVELS) {
668         energy_upgrade_string        += "[D]: + 200 kWh (";
669         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
670         energy_upgrade_string        += " K)\n";
671     }
672
673     else {
674         energy_upgrade_string += " * MAX LEVEL *  \n";
675     }
676
677     sf::Text energy_upgrade_text = sf::Text(
678         energy_upgrade_string,
679         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
680         16
681     );
682
683     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
684     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
685     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
686
687     this->render_window_ptr->draw(energy_upgrade_text);
688
689     return;
690 }   /* __drawUpgradeOptions() */
```

4.15.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
461 {
462     if (this->just_built) {
463         return;
464     }
465
466     switch (this->event_ptr->key.code) {
467         case (sf::Keyboard::U): {
468             this->__openUpgradeMenu();
469
470             break;
471         }
472
473
474         case (sf::Keyboard::W): {
475             if (this->production_menu_open) {
476                 this->dispatch_MWh++;
477
478                 if (this->dispatch_MWh > this->dispatchable_MWh) {
479                     this->dispatch_MWh = 0;
480                 }
481
482                 this->__computeProductionCosts();
483                 this->assets_manager_ptr->getSound("interface click")->play();
484             }
485
486             else if (this->upgrade_menu_open) {
487                 this->__upgradePowerCapacity();
488             }
```

```
489
490             break;
491         }
492
493
494         case (sf::Keyboard::S): {
495             if (this->production_menu_open) {
496                 this->dispatch_MWh--;
497
498                 if (this->dispatch_MWh < 0) {
499                     this->dispatch_MWh = this->dispatchable_MWh;
500                 }
501
502                 this->__computeProductionCosts();
503                 this->assets_manager_ptr->getSound("interface click")->play();
504             }
505
506             break;
507         }
508
509
510         case (sf::Keyboard::D): {
511             if (this->upgrade_menu_open) {
512                 this->__upgradeStorageCapacity();
513                 this->__computeProduction();
514                 this->__computeDispatch();
515             }
516
517             break;
518         }
519
520
521         default: {
522             // do nothing!
523
524             break;
525         }
526     }
527
528     return;
529 }   /* __handleKeyPressEvents() */
```

4.15.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
544 {
545     if (this->just_built) {
546         return;
547     }
548
549     switch (this->event_ptr->mouseButton.button) {
550         case (sf::Mouse::Left): {
551             //...
552
553             break;
554         }
555
556
557         case (sf::Mouse::Right): {
558             //...
559
560             break;
561         }
562
563
564         default: {
565             // do nothing!
566
567             break;
568         }
569     }
570
571     return;
572 }   /* __handleMouseButtonEvents() */
```

4.15.3.10 __repair()

```
void WindTurbine::__repair (
            void  )  [private], [virtual]
```

Helper method to repair the wind turbine.

Reimplemented from TileImprovement.

```
274 {
275     if (this->credits < WIND_TURBINE_BUILD_COST) {
276         std::cout « "Cannot repair wind turbine: insufficient credits (need "
277             « WIND_TURBINE_BUILD_COST « " K)" « std::endl;
278 
279         this->__sendInsufficientCreditsMessage();
280         return;
281     }
282 
283     TileImprovement :: __repair();
284 
285     this->just_upgraded = true;
286 
287     this->__sendCreditsSpentMessage(WIND_TURBINE_BUILD_COST);
288     this->__sendTileStateRequest();
289     this->__sendGameStateRequest();
290 
291     return;
292 }   /* __repair() */
```

4.15.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
705 {
706     Message improvement_state_message;
707 
708     improvement_state_message.channel = GAME_CHANNEL;
709     improvement_state_message.subject = "improvement state";
710 
711     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
712     improvement_state_message.int_payload["operation_maintenance_cost"] =
713         this->operation_maintenance_cost;
714 
715     this->message_hub_ptr->sendMessage(improvement_state_message);
716 
717     std::cout « "Improvement state message sent by " « this « std::endl;
718 
719     return;
720 }   /* __sendImprovementStateMessage() */
```

4.15.3.12 __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() */
```

4.15.3.13 __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     TileImprovement :: __repair();
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() */
```

4.15.3.14 advanceTurn()

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

Method to handle turn advance.

Reimplemented from TileImprovement.

```
925 {
926     //  1. send improvement state message
927     this->__sendImprovementStateMessage();
928 
929     //  2. update
930     this->__computeCapacityFactors();
931     this->update();
932 
933     //  3. handle start/stop
934     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
935         this->is_running = true;
936     }
937 
938     else if (this->is_running and (this->dispatch_MWh <= 0)) {
939         this->is_running = false;
940     }
941 
942     //  4. handle equipment health
943     if (this->is_running) {
944         this->health--;
945 
946         if (this->health <= 0) {
947             this->__breakdown();
948         }
949     }
950 
951     //  5. send tile state request (if selected)
952     if (this->is_selected) {
953         this->__sendTileStateRequest();
954     }
955 
956     return;
957 }   /* advanceTurn() */
```

4.15.3.15 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.

```
1048 {
1049     //  1. if just built, call base method and return
1050     if (this->just_built) {
1051         TileImprovement :: draw();
1052 
1053         return;
1054     }
1055 
1056 
1057     //  2. handle upgrade effects
1058     if (this->just_upgraded) {
1059         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1060             this->tile_improvement_sprite_animated[i].setColor(
1061                 sf::Color(
1062                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1063                     255,
1064                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1065                     255
1066                 )
1067             );
1068 
1069             this->tile_improvement_sprite_animated[i].setScale(
1070                 sf::Vector2f(
1071                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1072                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1073                 )
1074             );
1075         }
1076 
1077         this->upgrade_frame++;
1078     }
```

```
1079
1080     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1081         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1082             this->tile_improvement_sprite_animated[i].setColor(
1083                 sf::Color(255,255,255,255)
1084             );
1085
1086             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1087         }
1088
1089         this->just_upgraded = false;
1090         this->upgrade_frame = 0;
1091     }
1092
1093
1094     //  3. draw first element of animated sprite
1095     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1096
1097
1098     //  4. draw second element of animated sprite
1099     if (this->is_running) {
1100         this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1101     }
1102
1103     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1104
1105
1106     //  5. draw storage upgrades
1107     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1108         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1109     }
1110
1111
1112     //  6. handle dispatch illustration
1113     if (this->dispatch_MWh > 0) {
1114         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1115         this->__drawDispatch();
1116     }
1117
1118
1119     //  7. draw production menu
1120     if (this->production_menu_open) {
1121         this->render_window_ptr->draw(this->production_menu_backing);
1122         this->render_window_ptr->draw(this->production_menu_backing_text);
1123
1124         this->__drawProductionMenu();
1125     }
1126
1127
1128     //  8. draw upgrade menu
1129     if (this->upgrade_menu_open) {
1130         this->render_window_ptr->draw(this->upgrade_menu_backing);
1131         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1132
1133         this->__drawUpgradeOptions();
1134     }
1135
1136
1137     //  9. handle broken effects
1138     if (this->is_broken) {
1139         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1140             this->tile_improvement_sprite_animated[i].setColor(
1141                 sf::Color(
1142                     255,
1143                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1144                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1145                     255
1146                 )
1147             );
1148         }
1149     }
1150
1151     this->frame++;
1152     return;
1153 }   /* draw() */
```

4.15.3.16 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
837 {
838     //                  32 char x 17 line console "--------------------------------\n";
839     std::string options_substring                = "CAPACITY:      ";
840     options_substring                           += std::to_string(this->capacity_kW);
841     options_substring                           += " kW (level ";
842     options_substring                           += std::to_string(this->upgrade_level);
843     options_substring                           += ")\n";
844 
845     options_substring                           += "PRODUCTION:    ";
846     options_substring                           += std::to_string(this->production_MWh);
847     options_substring                           += " MWh\n";
848 
849     options_substring                           += "DISPATCHABLE:  ";
850     options_substring                           += std::to_string(this->dispatchable_MWh);
851     options_substring                           += " MWh\n";
852 
853     options_substring                           += "HEALTH:      ";
854     options_substring                           += std::to_string(this->health);
855     options_substring                           += "/100";
856 
857     if (this->health <= 0) {
858         options_substring                       += " ** BROKEN! **\n";
859     }
860 
861     else {
862         options_substring                       += "\n";
863     }
864 
865     options_substring                           += "                                \n";
866     options_substring                           += " **** WIND TURBINE OPTIONS **** \n";
867     options_substring                           += "                                \n";
868 
869     if (this->is_broken) {
870         options_substring                       += "     [R]:  REPAIR (";
871         options_substring                       += std::to_string(WIND_TURBINE_BUILD_COST);
872         options_substring                       += " K)\n";
873     }
874 
875     else {
876         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
877     }
878 
879     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
880     options_substring                           += "HOLD [P]:  SCRAP (";
881     options_substring                           += std::to_string(SCRAP_COST);
882     options_substring                           += " K)";
883 
884     return options_substring;
885 }   /* getTileOptionsSubstring() */
```

4.15.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
999 {
1000     TileImprovement :: processEvent();
1001 
1002     if (this->event_ptr->type == sf::Event::KeyPressed) {
1003         this->__handleKeyPressEvents();
1004     }
1005 
1006     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1007         this->__handleMouseButtonEvents();
1008     }
1009 
1010     return;
1011 }   /* processEvent() */
```

4.15.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1026 {
1027     TileImprovement :: processMessage();
1028
1029     //...
1030
1031     return;
1032 }   /* processMessage() */
```

4.15.3.19 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.

```
902 {
903     TileImprovement :: setIsSelected(is_selected);
904
905     if (this->is_running and this->is_selected) {
906         this->assets_manager_ptr->getSound("wind turbine running")->play();
907     }
908
909     return;
910 }   /* setIsSelected() */
```

4.15.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
972 {
973     std::cout « "WindTurbine :: update()" « std::endl;
974
975     this->__computeProduction();
976     this->__computeProductionCosts();
977     this->__computeDispatch();
978
979     if (this->is_selected) {
980         this->__sendTileStateRequest();
981     }
982
983     return;
984 }   /* 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.

4.15.4.9 rotor_drotation

```
double WindTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

- header/WindTurbine.h
- source/WindTurbine.cpp

Chapter 5

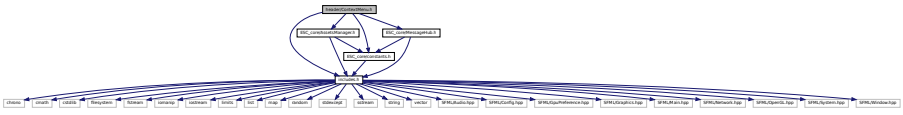
File Documentation

5.1 header/ContextMenu.h File Reference

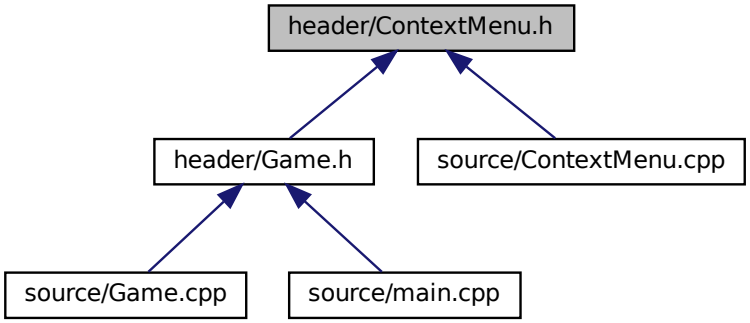
Header file for the ContextMenu class.

```
#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 };
```

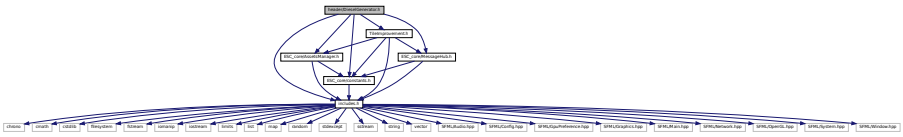
5.2 header/DieselGenerator.h File Reference

Header file for the DieselGenerator class.

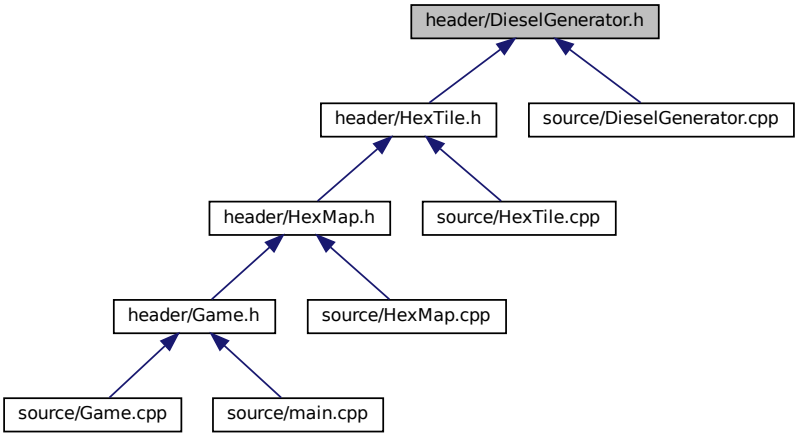
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
```

```
#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

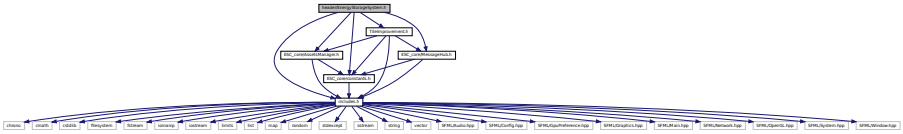
Header file for the DieselGenerator class.

5.3 header/EnergyStorageSystem.h File Reference

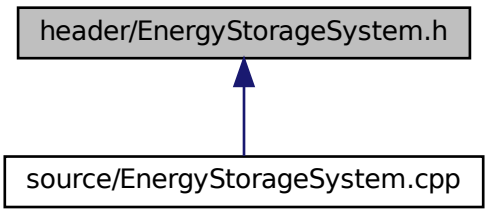
Header file for the EnergyStorageSystem class.

```
#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).

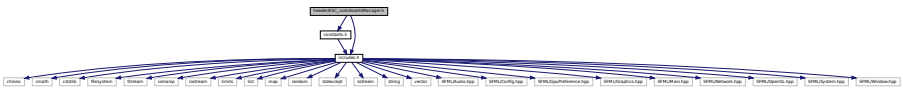
5.3.1 Detailed Description

Header file for the EnergyStorageSystem class.

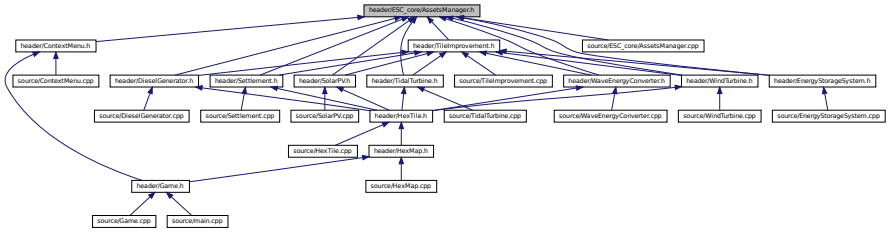
5.4 header/ESC_core/AssetsManager.h File Reference

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.

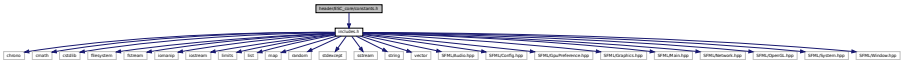
5.4.1 Detailed Description

Header file for the AssetsManager class.

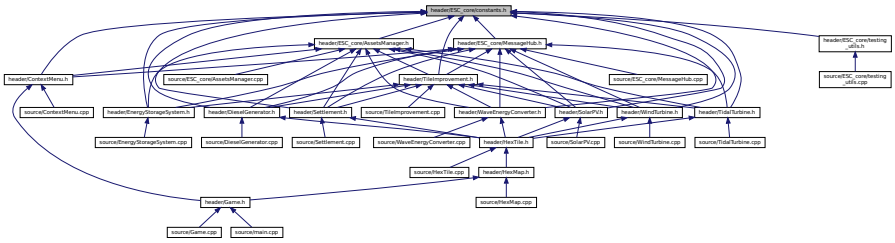
5.5 header/ESC_core/constants.h File Reference

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 = 160

  The cost of clearing a forest tile.
- const int CLEAR_MOUNTAINS_COST = 500

  The cost of clearing a mountains tile.
- const int CLEAR_PLAINS_COST = 80

  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 = 450

  The cost of building (or upgrading) a wind turbine in 100 kW increments.
- const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.222222

  The additional cost of building on water.
- const int SOLAR_PV_BUILD_COST = 350

  The cost of building (or upgrading) a solar PV array in 100 kW increments.
- const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.285714

  The additional cost of building on water.
- const int TIDAL_TURBINE_BUILD_COST = 550

  The cost of building (or upgrading) a tidal turbine in 100 kW increments.
- const int WAVE_ENERGY_CONVERTER_BUILD_COST = 850

  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 = 800

  The starting balance of credits.
- const double CREDITS_PER_MWH_SERVED = 1.125

  The number of credits (x1000) earned.
- const int EMISSIONS_LIFETIME_LIMIT_TONNES = 2000

  The lifetime limit on CO2-equivalent emissions (1 tonne CO2e ∼= 667 L diesel).
- 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 MEAN_POPULATION_GROWTH_RATE = 0.020

  The mean monthly population growth rate.
- const double STDEV_POPULATION_GROWTH_RATE = 0.005

  The standard deviation in monthly population growth rate.
- const double LITRES_DIESEL_PER_MWH_PRODUCTION = 375

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 = 1.75

  The cost of a litre of diesel.

- const double KG_CO2E_PER_LITRE_DIESEL = 3.16

  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 = 0.225

  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 (usually semi-diurnal, mostly driven by orbits of moon and sun).

- 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 = 160
```

The cost of clearing a forest tile.

5.5.3.3 CLEAR_MOUNTAINS_COST

```
const int CLEAR_MOUNTAINS_COST = 500
```

The cost of clearing a mountains tile.

5.5.3.4 CLEAR_PLAINS_COST

```
const int CLEAR_PLAINS_COST = 80
```

The cost of clearing a plains tile.

5.5.3.5 COST_PER_LITRE_DIESEL

```
const double COST_PER_LITRE_DIESEL = 1.75
```

The cost of a litre of diesel.

5.5.3.6 CREDITS_PER_MWH_SERVED

```
const double CREDITS_PER_MWH_SERVED = 1.125
```

The number of credits (x1000) earned.

5.5.3.7 DAILY_TIDAL_CAPACITY_FACTOR

```
const double DAILY_TIDAL_CAPACITY_FACTOR = 0.225
```

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 (usually semi-diurnal, mostly driven by orbits of moon and sun).

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 = 2000
```

The lifetime limit on CO2-equivalent emissions (1 tonne CO2e $\sim$= 667 L diesel).

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 = 3.16
```

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 = 375
```

The litres of diesel consumed in producing 1 MWh (assumes higher heating value and constant thermal efficiency of $\sim$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:

```
= {
    0.029, 0.061, 0.117,
    0.183, 0.228, 0.233,
    0.219, 0.185, 0.139,
    0.081, 0.040, 0.021
}
```

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 MEAN_POPULATION_GROWTH_RATE

```
const double MEAN_POPULATION_GROWTH_RATE = 0.020
```

The mean monthly population growth rate.

5.5.3.29 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

5.5.3.30 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 = 350
```

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 = 1.285714
```

The additional cost of building on water.

5.5.3.39 STARTING_CREDITS

```
const int STARTING_CREDITS = 800
```

The starting balance of credits.

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 STDEV_POPULATION_GROWTH_RATE

```
const double STDEV_POPULATION_GROWTH_RATE = 0.005
```

The standard deviation in monthly population growth rate.

5.5.3.46 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).

5.5.3.47 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 550
```

The cost of building (or upgrading) a tidal turbine in 100 kW increments.

5.5.3.48 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).

5.5.3.49 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.50 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.51 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).

5.5.3.52 WAVE_ENERGY_CONVERTER_BUILD_COST

```
const int WAVE_ENERGY_CONVERTER_BUILD_COST = 850
```

The cost of building (or upgrading) a wave energy converter in 100 kW increments.

5.5.3.53 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).

5.5.3.54 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).

5.5.3.55 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 450
```

The cost of building (or upgrading) a wind turbine in 100 kW increments.

5.5.3.56 WIND_TURBINE_WATER_BUILD_MULTIPLIER

```
const double WIND_TURBINE_WATER_BUILD_MULTIPLIER = 1.222222
```

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]

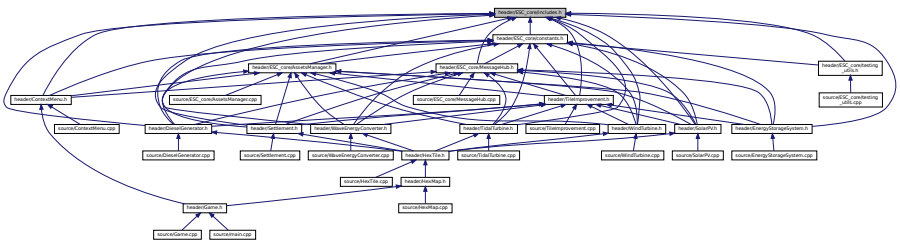
5.7 header/ESC_core/includes.h File Reference

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.

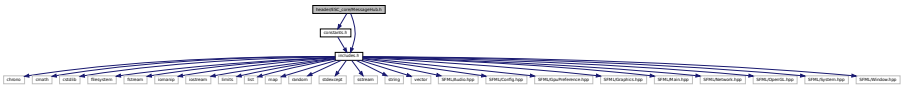
Ref: Gomila [2023]

5.8 header/ESC_core/MessageHub.h File Reference

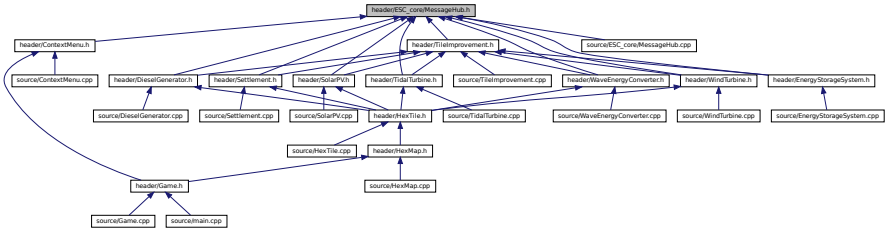
Header file for the MessageHub class.

```
#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

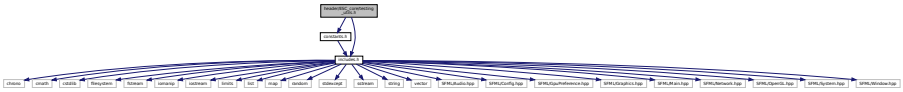
Header file for the MessageHub class.

5.9 header/ESC_core/testing_utils.h File Reference

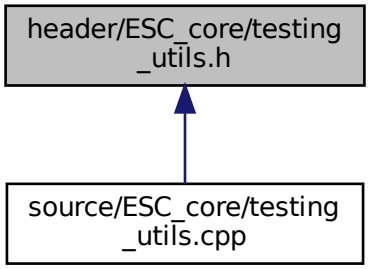
Header file for various testing utilities.

```
#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.
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

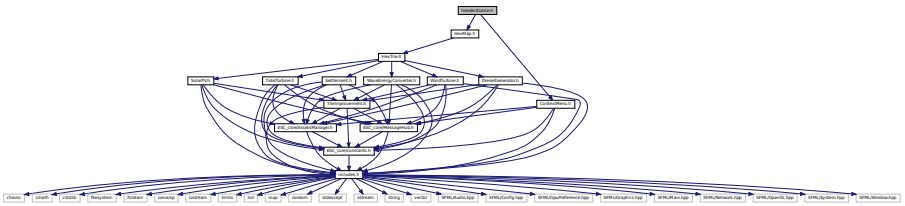
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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() */
```

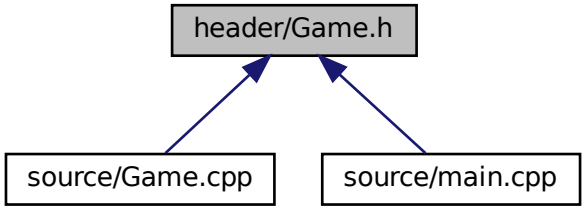
5.10 header/Game.h File Reference

```
#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.

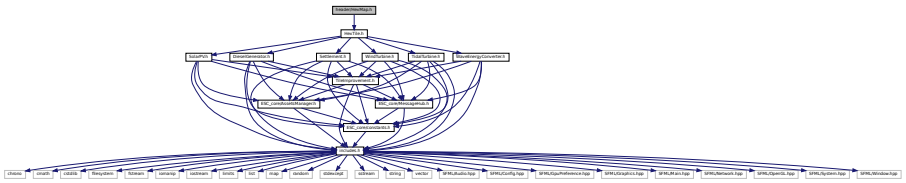
```
66                {
67      BUILD_SETTLEMENT,
68      SYSTEM_MANAGEMENT,
69      LOSS_EMISSIONS,
70      LOSS_DEMAND,
71      LOSS_CREDITS,
72      VICTORY,
73      N_GAME_PHASES
74  };  /* GamePhase */
```

5.11 header/HexMap.h File Reference

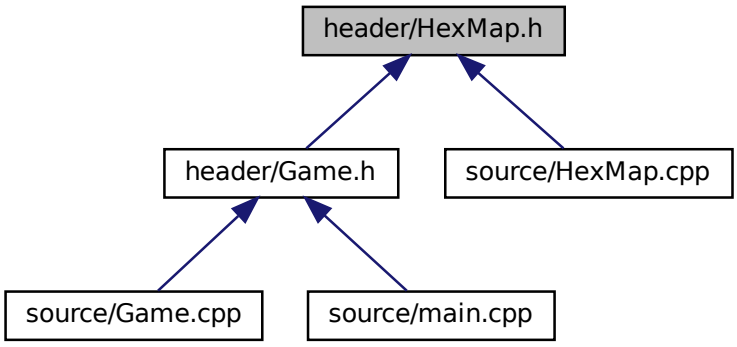
Header file for the HexMap class.

```
#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.

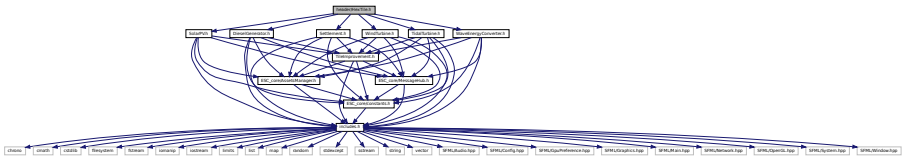
5.12 header/HexTile.h File Reference

Header file for the Game class.

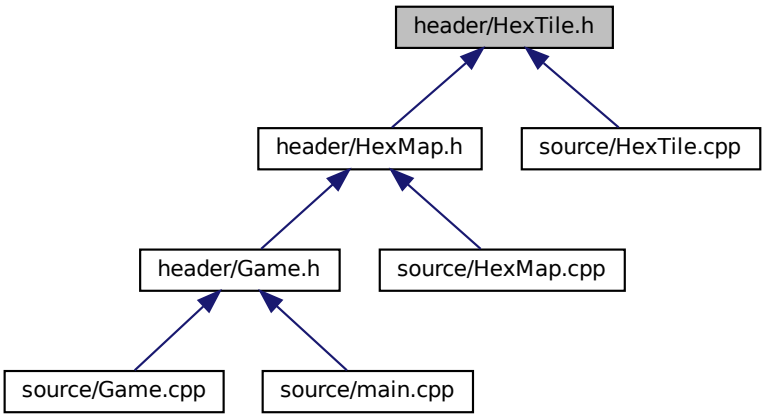
```
#include "DieselGenerator.h"
#include "Settlement.h"
#include "SolarPV.h"
#include "TidalTurbine.h"
#include "WaveEnergyConverter.h"
```

```
#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.

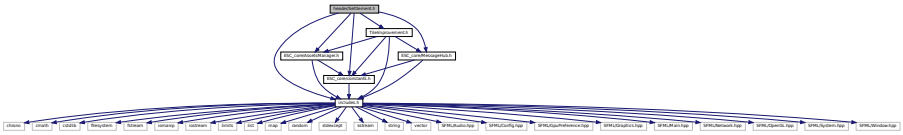
```
71              {
72     NONE_TYPE,
73     FOREST,
74     LAKE,
75     MOUNTAINS,
76     OCEAN,
77     PLAINS,
78     N_TILE_TYPES
79 }; /* TileType */
```

5.13 header/Settlement.h File Reference

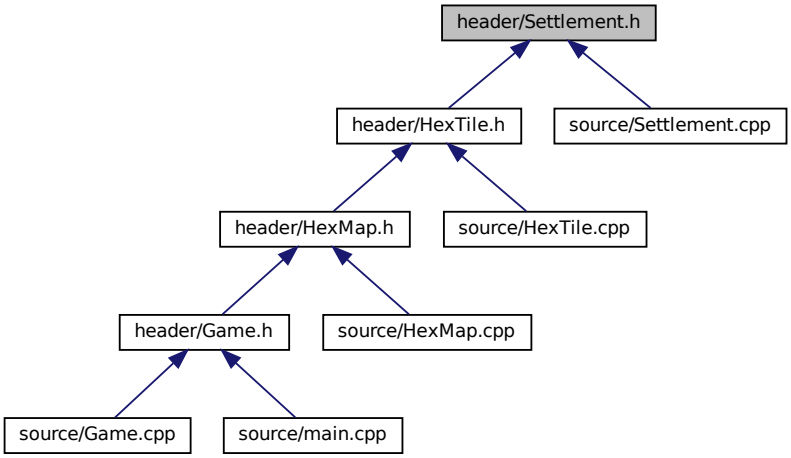
Header file for the Settlement class.

```
#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

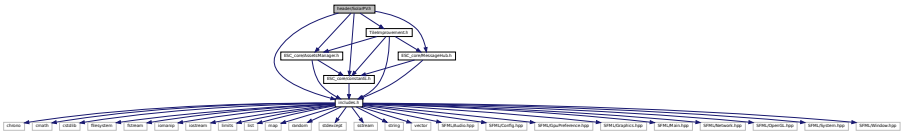
Header file for the Settlement class.

5.14 header/SolarPV.h File Reference

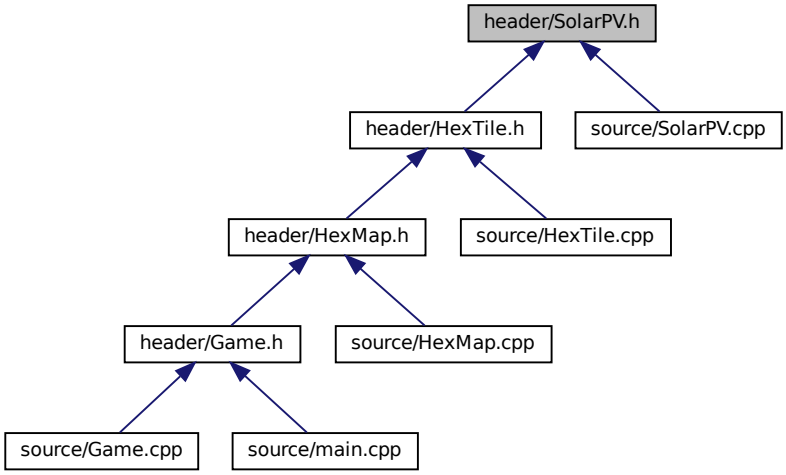
Header file for the SolarPV class.

```
#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

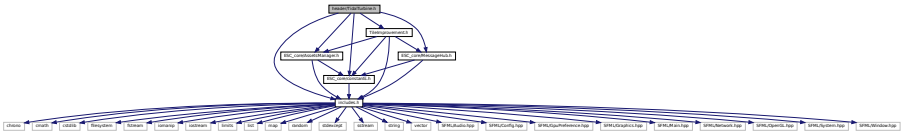
Header file for the SolarPV class.

5.15 header/TidalTurbine.h File Reference

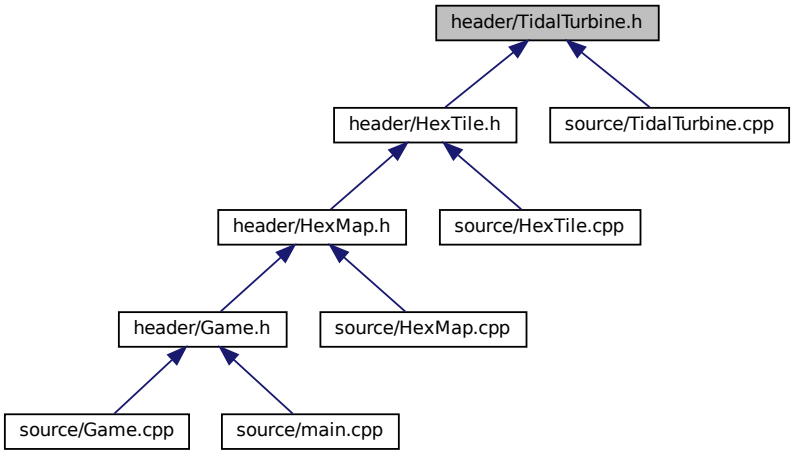
Header file for the TidalTurbine class.

```
#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

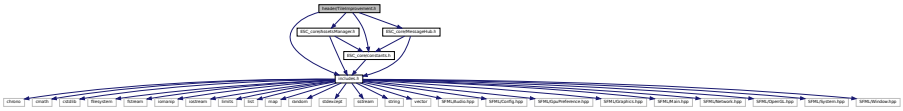
Header file for the TidalTurbine class.

5.16 header/TileImprovement.h File Reference

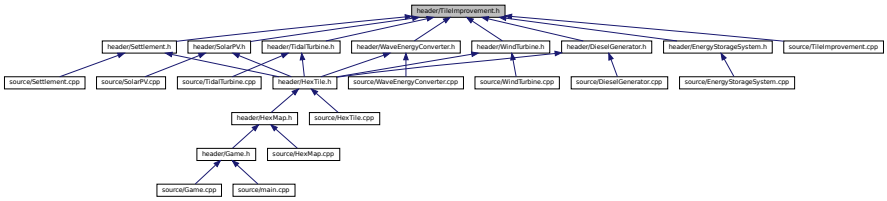
Header file for the TileImprovement class.

```
#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 , 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.
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     N_TILE_IMPROVEMENT_TYPES
76 };  /* TileImprovementType */
```

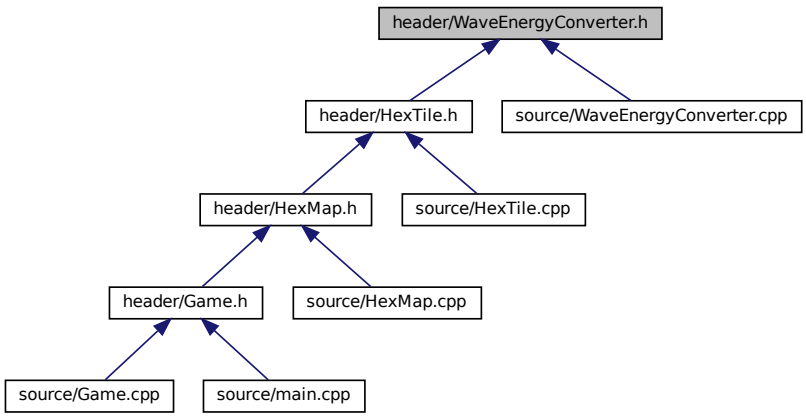
5.17 header/WaveEnergyConverter.h File Reference

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

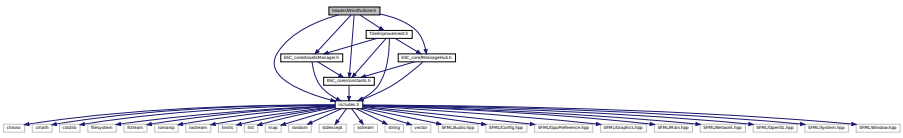
Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

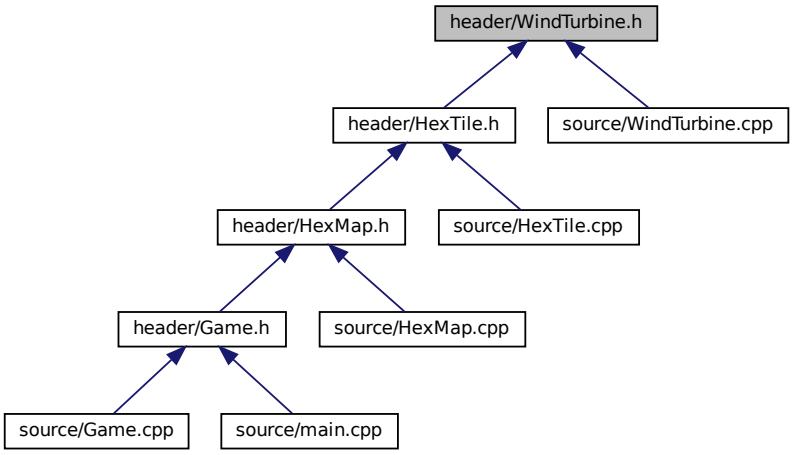
Header file for the WindTurbine class.

```
#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).

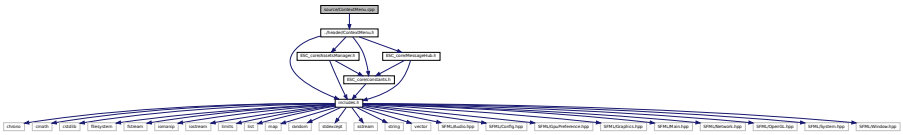
5.18.1 Detailed Description

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

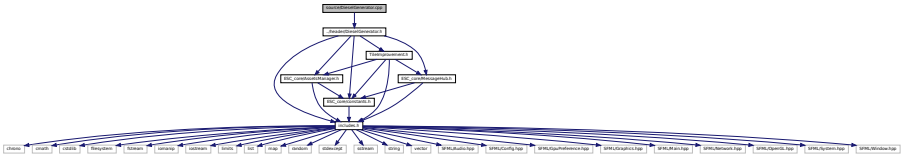
Implementation file for the ContextMenu class.

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

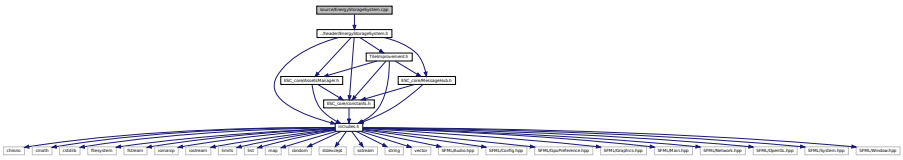
Implementation file for the DieselGenerator class.

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

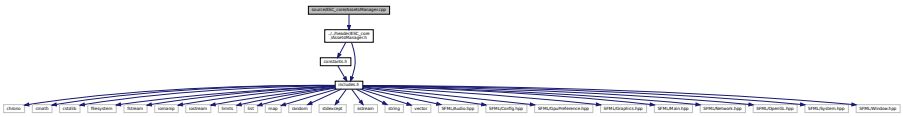
Implementation file for the EnergyStorageSystem class.

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

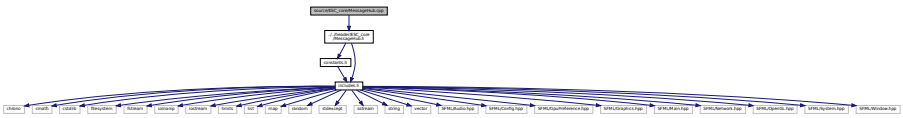
Implementation file for the AssetsManager class.

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.

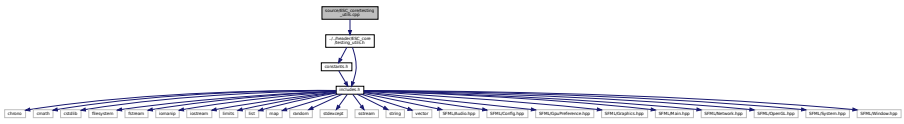
A class which acts as a central hub for inter-object message traffic.

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

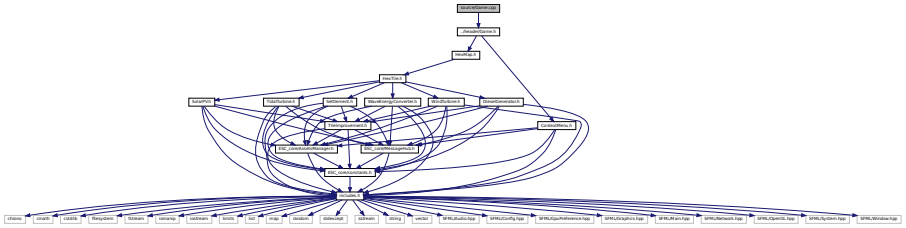
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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

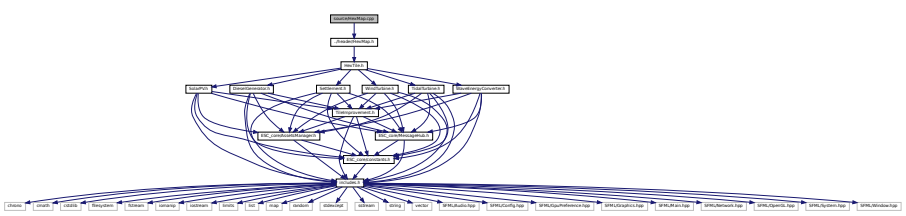
Implementation file for the Game class.

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

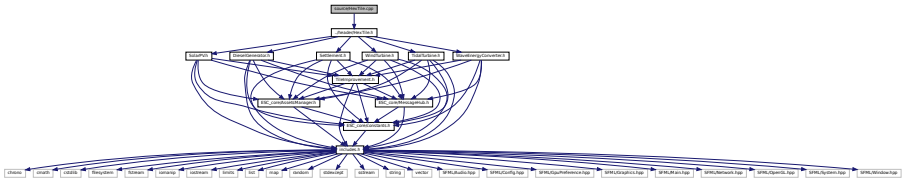
Implementation file for the HexMap class.

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

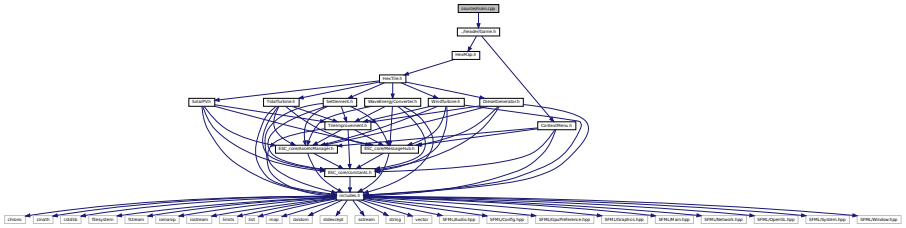
Implementation file for the HexTile class.

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.

```
339 {
340     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
341         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
342         "Road To Zero"
343     );
344 
345     return render_window_ptr;
346 }   /* 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     assets_manager_ptr->loadSound(
296         "assets/audio/samples/mixkit-fantasy-game-success-notification-270_MixkitFree.ogg",
297         "victory"
298     );
299
300     assets_manager_ptr->loadSound(
301         "assets/audio/samples/mixkit-player-losing-or-failing-2042_MixkitFree.ogg",
302         "loss"
303     );
304
305
306     //  4. load tracks
307     assets_manager_ptr->loadTrack(
308         "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
309         "Tree Star Moon - Dobranoc"
310     );
311
312     assets_manager_ptr->loadTrack(
313         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
314         "Tree Star Moon - Lighthouse"
315     );
316
317     assets_manager_ptr->loadTrack(
318         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
319         "Tree Star Moon - Sky Farm"
320     );
321
322     return;
323 }   /* loadAssets() */
```

5.28.2.3 main()

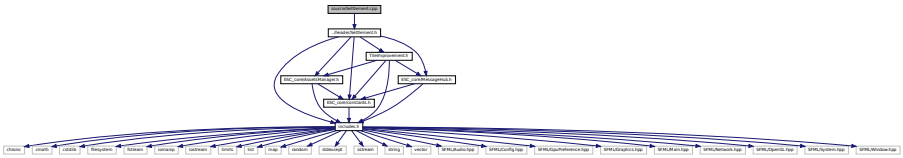
```
int main (
              int argc,
              char ** argv )
355 {
356     //  1. load assets
357     AssetsManager assets_manager;
358     loadAssets(&assets_manager);
359
360     //  2. construct render window
361     sf::RenderWindow* render_window_ptr = constructRenderWindow();
```

```
362
363     //  3. start game loop
364     bool quit_game = false;
365     assets_manager.playTrack();
366
367     while (not quit_game) {
368         Game game(render_window_ptr, &assets_manager);
369         quit_game = game.run();
370     }
371
372     //  4. clean up
373     render_window_ptr->close();
374     delete render_window_ptr;
375
376     return 0;
377 }   /* 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

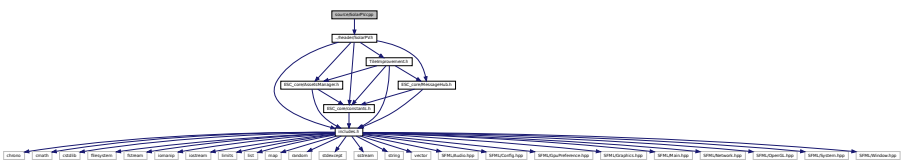
Implementation file for the Settlement class.

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

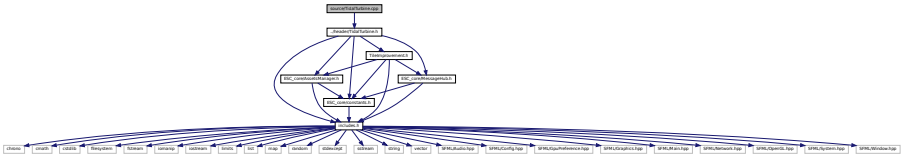
Implementation file for the SolarPV class.

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

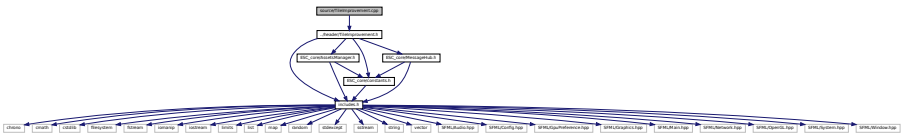
Implementation file for the TidalTurbine class.

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

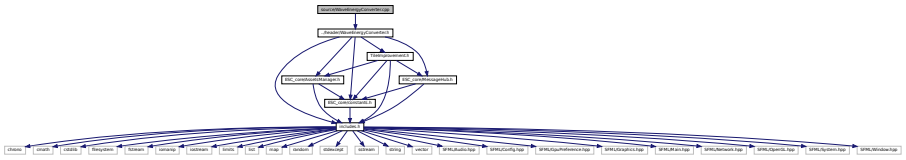
Implementation file for the TileImprovement class.

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

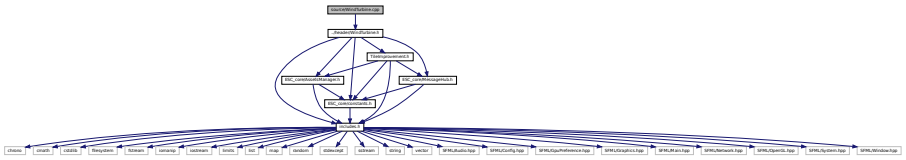
Implementation file for the WaveEnergyConverter class.

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/. 299

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 298

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 40, 54, 121, 174, 184, 201, 221, 243, 261

Index