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:

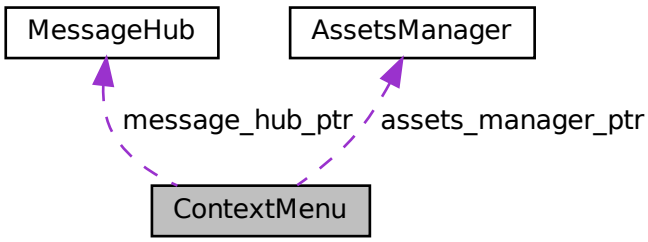
- header/ESC_core/AssetsManager.h
- source/ESC_core/AssetsManager.cpp

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]:    TOGGLE 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:

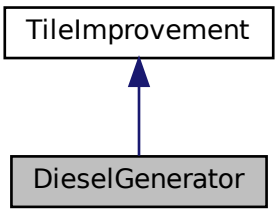
- header/ContextMenu.h
- source/ContextMenu.cpp

4.3 DieselGenerator Class Reference

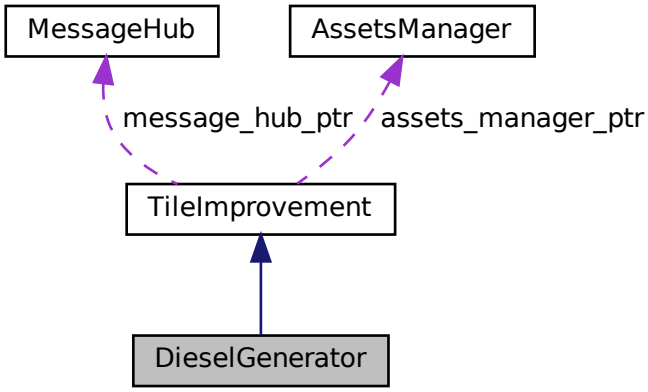
A settlement class (child class of TileImprovement).

```
#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.

```
508   :
509 TileImprovement(
510     position_x,
511     position_y,
512     tile_resource,
513     event_ptr,
514     render_window_ptr,
515     assets_manager_ptr,
516     message_hub_ptr
517 )
518 {
519     //  1. set attributes
520
521     //  1.1. private
522     //...
523
524     //  1.2. public
525     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
526
527     this->is_running = false;
528
529     this->health = 100;
530
531     this->capacity_kW = 200;
532     this->upgrade_level = 1;
533
534     this->production_MWh = 0;
535     this->max_production_MWh = 144;
536
537     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
538     this->smoke_dx = 5 * SECONDS_PER_FRAME;
539     this->smoke_dy = -10 * SECONDS_PER_FRAME;
540     this->smoke_prob = 16 * SECONDS_PER_FRAME;
541
542     this->smoke_sprite_list = {};
543
544     this->fuel_cost = 0;
545     this->emissions_tonnes_CO2e = 0;
546
547     this->tile_improvement_string = "DIESEL GEN";
548
549     this->__setUpTileImprovementSpriteAnimated();
550
551     std::cout « "DieselGenerator constructed at " « this « std::endl;
552
553     return;
```

```
554 }   /* DieselGenerator() */
```

4.3.2.2 ~DieselGenerator()

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

Destructor for the DieselGenerator class.

```
935 {
936     std::cout « "DieselGenerator at " « this « " destroyed" « std::endl;
937
938     return;
939 }   /* ~DieselGenerator() */
```

4.3.3 Member Function Documentation

4.3.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
270 {
271     TileImprovement :: __breakdown();
272
273     this->production_MWh = 0;
274     this->fuel_cost = 0;
275     this->operation_maintenance_cost = 0;
276     this->emissions_tonnes_CO2e = 0;
277
278     return;
279 }   /* __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 performance_factor = this->__getPerformanceFactor();
237
238     if (performance_factor > 0) {
239         litres_diesel /= performance_factor;
240     }
241
242     double fuel_cost = (litres_diesel * COST_PER_LITRE_DIESEL) / 1000;
243     this->fuel_cost = round(fuel_cost);
244
245     double emissions_tonnes_CO2e = (litres_diesel * KG_CO2E_PER_LITRE_DIESEL) / 1000;
246     this->emissions_tonnes_CO2e = round(emissions_tonnes_CO2e);
247
248     double operation_maintenance_cost =
249         (this->production_MWh * DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
250     this->operation_maintenance_cost = round(operation_maintenance_cost);
251
252     this->__sendTileStateRequest();
253
254     return;
255 }   /* __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.

```
327 {
328     if (this->just_built) {
329         return;
330     }
331 
```

```
332
333     switch (this->event_ptr->key.code) {
334         case (sf::Keyboard::U): {
335             this->__upgrade();
336
337             break;
338         }
339
340
341         case (sf::Keyboard::W): {
342             if (this->production_menu_open) {
343                 this->production_MWh++;
344
345                 if (this->production_MWh > this->max_production_MWh) {
346                     this->production_MWh = 0;
347                 }
348
349                 this->__computeProductionCosts();
350                 this->assets_manager_ptr->getSound("interface click")->play();
351             }
352
353             break;
354         }
355
356
357         case (sf::Keyboard::S): {
358             if (this->production_menu_open) {
359                 this->production_MWh--;
360
361                 if (this->production_MWh < 0) {
362                     this->production_MWh = this->max_production_MWh;
363                 }
364
365                 this->__computeProductionCosts();
366                 this->assets_manager_ptr->getSound("interface click")->play();
367             }
368
369             break;
370         }
371
372
373         default: {
374             // do nothing!
375
376             break;
377         }
378     }
379
380
381     return;
382 }   /* __handleKeyPressEvents() */
```

4.3.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
397 {
398     if (this->just_built) {
399         return;
400     }
401
402     switch (this->event_ptr->mouseButton.button) {
403         case (sf::Mouse::Left): {
404             //...
405
406             break;
407         }
408
409
410         case (sf::Mouse::Right): {
411             //...
412
413             break;
414         }
415
416
```

```
417         default: {
418             // do nothing!
419 
420             break;
421         }
422     }
423 
424     return;
425 }   /* __handleMouseButtonEvents() */
```

4.3.3.6 __repair()

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

Helper method to repair the diesel generator.

Reimplemented from TileImprovement.

```
294 {
295     if (this->credits < DIESEL_GENERATOR_BUILD_COST) {
296         std::cout « "Cannot repair diesel generator: insufficient credits (need "
297             « DIESEL_GENERATOR_BUILD_COST « " K)" « std::endl;
298 
299         this->__sendInsufficientCreditsMessage();
300         return;
301     }
302 
303     TileImprovement :: __repair();
304 
305     this->just_upgraded = true;
306 
307     this->__sendCreditsSpentMessage(DIESEL_GENERATOR_BUILD_COST);
308     this->__sendTileStateRequest();
309     this->__sendGameStateRequest();
310 
311     return;
312 }   /* __repair() */
```

4.3.3.7 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
440 {
441     Message improvement_state_message;
442 
443     improvement_state_message.channel = GAME_CHANNEL;
444     improvement_state_message.subject = "improvement state";
445 
446     improvement_state_message.int_payload["dispatch_MWh"] = this->production_MWh;
447     improvement_state_message.int_payload["fuel_cost"] = this->fuel_cost;
448     improvement_state_message.int_payload["operation_maintenance_cost"] =
449         this->operation_maintenance_cost;
450     improvement_state_message.int_payload["emissions_tonnes_CO2e"] =
451         this->emissions_tonnes_CO2e;
452 
453     this->message_hub_ptr->sendMessage(improvement_state_message);
454 
455     std::cout « "Improvement state message sent by " « this « std::endl;
456 
457     return;
458 }   /* __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 += 200;
203     this->upgrade_level++;
204
205     this->production_MWh = 0;
206     this->max_production_MWh += 144;
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.

```
664 {
665     //  1. send improvement state message
666     this->__sendImprovementStateMessage();
667
668     //  2. handle start/stop
669     if ((not this->is_running) and (this->production_MWh > 0)) {
670         this->is_running = true;
671         this->assets_manager_ptr->getSound("diesel start")->play();
672     }
673
674     else if (this->is_running and (this->production_MWh <= 0)) {
675         this->is_running = false;
676         this->tile_improvement_sprite_animated[1].setScale(sf::Vector2f(1, 1));
677     }
678
679     //  3. handle equipment health and breakdowns
680     if (this->is_running) {
681         this->health--;
682
683         if (this->health <= 50) {
684             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
685
686             if ((double)rand() / RAND_MAX <= breakdown_prob) {
687                 this->health = 0;
688             }
689         }
690
691         if (this->health <= 0) {
692             this->__breakdown();
693         }
694     }
695
696     //  4. send tile state request (if selected)
697     if (this->is_selected) {
698         this->__sendTileStateRequest();
699     }
700
701     return;
702 }   /* 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.

```
766 {
767     //  1. if just built, call base method and return
768     if (this->just_built) {
769         TileImprovement :: draw();
770
771         return;
772     }
773
774     //  2. handle upgrade effects
775     if (this->just_upgraded) {
776         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
777             this->tile_improvement_sprite_animated[i].setColor(
778                 sf::Color(
779                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
780                     255,
781                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
782                     255
```

```
783                 )
784             );
785 
786             this->tile_improvement_sprite_animated[i].setScale(
787                 sf::Vector2f(
788                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
789                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
790                 )
791             );
792         }
793 
794         this->upgrade_frame++;
795     }
796 
797     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
798         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
799             this->tile_improvement_sprite_animated[i].setColor(
800                 sf::Color(255,255,255,255)
801             );
802 
803             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
804         }
805 
806         this->just_upgraded = false;
807         this->upgrade_frame = 0;
808     }
809 
810 
811     //  3. draw first element of animated sprite
812     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
813 
814 
815     //  4. draw second element of animated sprite
816     double move_x = 0;
817     double move_y = 0;
818 
819     if (this->is_running) {
820         this->tile_improvement_sprite_animated[1].setScale(
821             sf::Vector2f(
822                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2),
823                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2)
824             )
825         );
826 
827         move_x = 1 * ((double)rand() / RAND_MAX) - 0.5;
828         move_y = 1 * ((double)rand() / RAND_MAX) - 0.5;
829 
830         this->tile_improvement_sprite_animated[1].move(move_x, move_y);
831     }
832 
833     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
834 
835     if (this->is_running) {
836         this->tile_improvement_sprite_animated[1].move(-1 * move_x, -1 * move_y);
837     }
838 
839 
840     //  5. draw smoke effects
841     if (this->is_running) {
842         if ((double)rand() / RAND_MAX < smoke_prob) {
843             this->smoke_sprite_list.push_back(
844                 sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
845             );
846 
847             this->smoke_sprite_list.back().setOrigin(
848                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
849                 this->smoke_sprite_list.back().getLocalBounds().height / 2
850             );
851 
852             this->smoke_sprite_list.back().setPosition(
853                 this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
854                 this->position_y - 33
855             );
856         }
857     }
858 
859     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
860 
861     double alpha = 255;
862 
863     while (iter != this->smoke_sprite_list.end()) {
864         this->render_window_ptr->draw(*iter);
865 
866         alpha = (*iter).getColor().a;
867 
868         alpha -= this->smoke_da;
869 
```

```
870         if (alpha <= 0) {
871             iter = this->smoke_sprite_list.erase(iter);
872             continue;
873         }
874 
875         (*iter).setColor(sf::Color(255, 255, 255, alpha));
876 
877         (*iter).move(
878             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
879             this->smoke_dy
880         );
881 
882         (*iter).rotate(((double)rand() / RAND_MAX));
883 
884         iter++;
885     }
886 
887 
888     //  6. handle dispatch illustration
889     if (this->production_MWh > 0) {
890         this->dispatch_text.setString(std::to_string(this->production_MWh));
891         this->__drawDispatch();
892     }
893 
894 
895     //  7. draw production menu
896     if (this->production_menu_open) {
897         this->render_window_ptr->draw(this->production_menu_backing);
898         this->render_window_ptr->draw(this->production_menu_backing_text);
899 
900         this->__drawProductionMenu();
901     }
902 
903 
904     //  8. handle broken effects
905     if (this->is_broken) {
906         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
907             this->tile_improvement_sprite_animated[i].setColor(
908                 sf::Color(
909                     255,
910                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
911                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
912                     255
913                 )
914             );
915         }
916     }
917 
918     this->frame++;
919     return;
920 }   /* 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.

```
571 {
572     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
573 
574     //                  32 char x 17 line console "--------------------------------\n";
575     std::string options_substring                = "CAPACITY:    ";
576     options_substring                           += std::to_string(this->capacity_kW);
577     options_substring                           += " kW (level ";
578     options_substring                           += std::to_string(this->upgrade_level);
579     options_substring                           += ")\n";
580 
```

```
581     options_substring                           += "PRODUCTION:  ";
582     options_substring                           += std::to_string(this->production_MWh);
583     options_substring                           += " MWh (MAX ";
584     options_substring                           += std::to_string(this->max_production_MWh);
585     options_substring                           += ")\n";
586 
587     options_substring                           += "HEALTH:      ";
588     options_substring                           += std::to_string(this->health);
589     options_substring                           += "/100";
590 
591     if (this->health <= 0) {
592         options_substring                       += " ** BROKEN! **\n";
593     }
594 
595     else {
596         options_substring                       += "\n";
597     }
598 
599     options_substring                           += "                                \n";
600     options_substring                           += "  **** DIESEL GEN OPTIONS ****  \n";
601     options_substring                           += "                                \n";
602 
603     if (this->is_broken) {
604         options_substring                       += "     [R]:  REPAIR (";
605         options_substring                       += std::to_string(DIESEL_GENERATOR_BUILD_COST);
606         options_substring                       += " K)\n";
607     }
608 
609     else {
610         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
611     }
612 
613     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
614         options_substring                           += "     [U]:  + 200 kW (";
615         options_substring                           += std::to_string(upgrade_cost);
616         options_substring                           +=" K)\n";
617     }
618 
619     options_substring                           += "HOLD [P]:  SCRAP (";
620     options_substring                           += std::to_string(SCRAP_COST);
621     options_substring                           += " K)";
622 
623     return options_substring;
624 }   /* getTileOptionsSubstring() */
```

4.3.3.13 processEvent()

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

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

Reimplemented from TileImprovement.

```
717 {
718     TileImprovement :: processEvent();
719 
720     if (this->event_ptr->type == sf::Event::KeyPressed) {
721         this->__handleKeyPressEvents();
722     }
723 
724     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
725         this->__handleMouseButtonEvents();
726     }
727 
728     return;
729 }   /* processEvent() */
```

4.3.3.14 processMessage()

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

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

Reimplemented from TileImprovement.

```
744 {
745     TileImprovement :: processMessage();
746
747     //...
748
749     return;
750 }   /* 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.

```
641 {
642     TileImprovement :: setIsSelected(is_selected);
643
644     if (this->is_running and this->is_selected) {
645         this->assets_manager_ptr->getSound("diesel running")->play();
646     }
647
648     return;
649 }   /* 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:

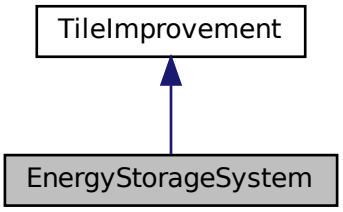
- header/DieselGenerator.h
- source/DieselGenerator.cpp

4.4 EnergyStorageSystem Class Reference

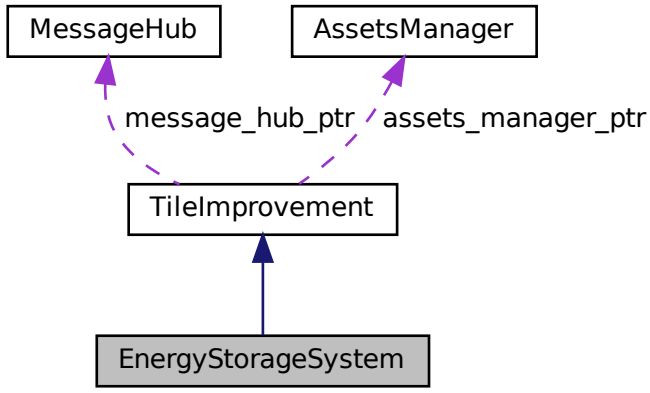
A settlement class (child class of TileImprovement).

```
#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:

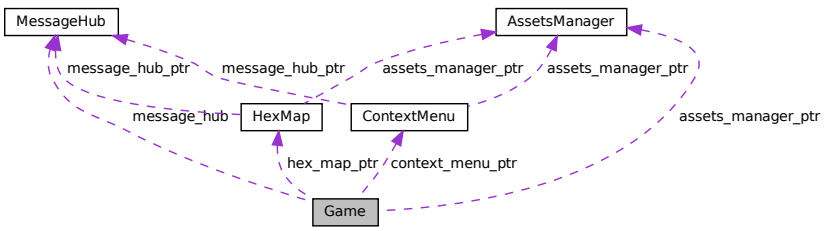
- header/EnergyStorageSystem.h
- source/EnergyStorageSystem.cpp

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 *, bool)

  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.
- bool draw_turn_advance_banner

  A boolean indicating whether or not to draw the turn advance banner.
- bool increase_turn_advance_alpha

  A boolean which indicates whether the turn advance alpha is increasing or decreasing.
- bool transition_from_title

  A boolean which indicates if construction follows a title transition.
- size_t tutorial_page

  Index for which page of the tutorial to show.
- std::string tutorial_string

  A string representation of the current tutorial page.
- sf::Text tutorial_text

  A text representation (drawable) of the tutorial page.
- 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.
- double turn_advance_alpha

  The alpha value for the turn advance banner.
- 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 or tutorial 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.
- sf::RectangleShape fade_rectangle

  A fading rectangle (for smooth transition from title to game).
- 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 __toggleTutorial (void)

  Helper method to handle toggling the tutorial on and off.
- void __incrementTutorial (void)

  Helper method to increment tutorial page (with wrap around).
- void __decrementTutorial (void)

  Helper method to decrement tutorial page (with wrap around).
- 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 __drawTurnAdvanceBanner (void)

  Helper method to draw turn advance banner.
- void __drawTutorial (void)

  Helper method to draw tutorial text.
- 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,
            bool transition_from_title )
```

Constructor for the Game class.

Parameters

render_window_ptr	Pointer to the render window.
assets_manager_ptr	Pointer to the assets manager.
transition_from_title	Boolean which indicates if this construction is following the title transition.

```
1748 {
1749     //  1. set attributes
1750
1751     //  1.1. private
1752     this->render_window_ptr = render_window_ptr;
1753
1754     this->assets_manager_ptr = assets_manager_ptr;
1755
1756     //  1.2. public
1757     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
1758
1759     this->quit_game = false;
1760     this->game_loop_broken = false;
1761     this->show_frame_clock_overlay = false;
```

```
1762     this->check_terminating_conditions = false;
1763     this->show_tutorial = true;
1764     this->turn_end = false;
1765     this->draw_turn_advance_banner = false;
1766     this->increase_turn_advance_alpha = true;
1767     this->transition_from_title = transition_from_title;
1768 
1769     this->tutorial_page = 0;
1770     this->tutorial_string = TUTORIAL_PAGES[this->tutorial_page];
1771 
1772     this->tutorial_text.setFont(
1773         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
1774     );
1775     this->tutorial_text.setCharacterSize(16);
1776     this->tutorial_text.setFillColor(MONOCHROME_TEXT_GREEN);
1777     this->tutorial_text.setPosition(GAME_WIDTH - 400 + 64, 64);
1778 
1779     this->frame = 0;
1780     this->time_since_start_s = 0;
1781 
1782     this->message_deadlock = false;
1783     this->message_deadlock_frame = 0;
1784 
1785     double seconds_since_epoch = time(NULL);
1786     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
1787 
1788     this->year = 1970 + (int)years_since_epoch;
1789     this->month = 0;
1790 
1791     this->population = 0;
1792     this->credits = STARTING_CREDITS;
1793     this->demand_MWh = 0;
1794     this->cumulative_emissions_tonnes = 0;
1795 
1796     this->past_demand_MWh = 0;
1797     this->turn_advance_alpha = 0;
1798 
1799     this->demand_vec_MWh.resize(30, 0);
1800 
1801     this->demand_served_MWh = 0;
1802     this->demand_remaining_MWh = 0;
1803     this->overproduction_MWh = 0;
1804     this->turn_fuel_cost = 0;
1805     this->turn_operation_maintenance_cost = 0;
1806     this->turn_emissions_tonnes = 0;
1807 
1808     this->overproduction_penalty = 0;
1809     this->dispatch_income = 0;
1810     this->net_credit_flow = 0;
1811 
1812     this->consecutive_zero_emissions_months = 0;
1813 
1814     this->substring_idx = 0;
1815     this->turn_summary_string = "";
1816 
1817     this->turn_summary_text.setFont(
1818         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
1819     );
1820     this->turn_summary_text.setCharacterSize(16);
1821     this->turn_summary_text.setFillColor(MONOCHROME_TEXT_GREEN);
1822     this->turn_summary_text.setPosition(GAME_WIDTH - 400 + 64, 64);
1823 
1824     this->fade_rectangle.setSize(sf::Vector2f(GAME_WIDTH, GAME_HEIGHT));
1825     this->fade_rectangle.setFillColor(sf::Color(0, 0, 0, 255));
1826 
1827     this->hex_map_ptr = new HexMap(
1828         6,
1829         &(this->event),
1830         this->render_window_ptr,
1831         this->assets_manager_ptr,
1832         &(this->message_hub)
1833     );
1834 
1835     this->context_menu_ptr = new ContextMenu(
1836         &(this->event),
1837         this->render_window_ptr,
1838         this->assets_manager_ptr,
1839         &(this->message_hub)
1840     );
1841 
1842     //  2. add message channel(s)
1843     this->message_hub.addChannel(GAME_CHANNEL);
1844     this->message_hub.addChannel(GAME_STATE_CHANNEL);
1845 
1846     this->__sendGameStateMessage();
1847 
1848     std::cout « "Game constructed at " « this « std::endl;
```

```
1849
1850     return;
1851 }   /* Game() */
```

4.5.2.2 ~Game()

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

Destructor for the Game class.

```
1978 {
1979     //  1. clean up attributes
1980     delete this->hex_map_ptr;
1981     delete this->context_menu_ptr;
1982
1983     std::cout « "Game at " « this « " destroyed" « std::endl;
1984
1985     return;
1986 }   /* ~Game() */
```

4.5.3 Member Function Documentation

4.5.3.1 __advanceTurn()

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

Helper method to advance turn.

```
170 {
171     //  1. advance turn, raise turn end flag
172     this->turn++;
173     this->turn_end = true;
174
175     //  2. reset turn summary attributes
176     this->demand_served_MWh = 0;
177     this->demand_remaining_MWh = 0;
178     this->overproduction_MWh = 0;
179     this->turn_fuel_cost = 0;
180     this->turn_operation_maintenance_cost = 0;
181     this->turn_emissions_tonnes = 0;
182
183     this->overproduction_penalty = 0;
184     this->dispatch_income = 0;
185     this->net_credit_flow = 0;
186
187     //  3. advance month/year
188     this->month++;
189     if (this->month > 12) {
190         this->year++;
191         this->month = 1;
192     }
193
194     //  4. update population
195     if (this->turn == 1) {
196         this->population = STARTING_POPULATION;
197     }
198
199     else {
200         this->__updatePopulation();
201     }
202
203     //  5. update demand
204     this->__computeCurrentDemand();
205
206     //  6. send turn advance message
207     this->__sendTurnAdvanceMessage();
208     this->__sendGameStateMessage();
209
210 }   /* __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     //  1. loss emissions
96     if (this->cumulative_emissions_tonnes >= EMISSIONS_LIFETIME_LIMIT_TONNES) {
97         this->assets_manager_ptr->getSound("loss")->play();
98         this->game_phase = GamePhase :: LOSS_EMISSIONS;
99     }
100
101     //  2. loss demand
102     else if (this->demand_remaining_MWh > 0) {
103         this->assets_manager_ptr->getSound("loss")->play();
104         this->game_phase = GamePhase :: LOSS_DEMAND;
105     }
106
107     //  3. loss credits
108     else if (this->credits < 0) {
109         this->assets_manager_ptr->getSound("loss")->play();
110         this->game_phase = GamePhase :: LOSS_CREDITS;
111     }
112
113     //  4. victory
114     else if (
115         (this->population >= 1000) and
116         (this->consecutive_zero_emissions_months >= 12)
117     ) {
118         this->assets_manager_ptr->getSound("victory")->play();
119         this->game_phase = GamePhase :: VICTORY;
120     }
121
122     //  5. send game state message
123     //this->__sendGameStateMessage();
124
125     return;
126 }   /* __checkTerminatingConditions() */
```

4.5.3.3 __computeCurrentDemand()

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

Helper method to compute current energy demand.

```
225 {
226     this->past_demand_MWh = this->demand_MWh;
227
228     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
229     std::default_random_engine generator(seed);
230
231     std::normal_distribution<double> normal_dist(
232         MEAN_DAILY_DEMAND_RATIOS[this->month - 1],
233         STDEV_DAILY_DEMAND_RATIOS[this->month - 1]
234     );
235
236     double demand_MWh = 0;
237
238     for (int i = 0; i < 30; i++) {
239         this->demand_vec_MWh[i] =
240             normal_dist(generator) * MAXIMUM_DAILY_DEMAND_PER_CAPITA * this->population;
241
242         demand_MWh += this->demand_vec_MWh[i];
243     }
244
245     this->demand_MWh = round(demand_MWh);
246
247     return;
248 }   /* __computeCurrentDemand() */
```

4.5.3.4 __decrementTutorial()

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

Helper method to decrement tutorial page (with wrap around).

```
322 {
323     if (this->tutorial_page == 0) {
324         this->tutorial_page = TUTORIAL_PAGES.size() - 1;
325     }
326
327     else {
328         this->tutorial_page--;
329     }
330
331     this->tutorial_string = TUTORIAL_PAGES[this->tutorial_page];
332     this->substring_idx = 0;
333
334     this->assets_manager_ptr->getSound("interface click")->play();
335
336     return;
337 }   /* __decrementTutorial() */
```

4.5.3.5 __draw()

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

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

```
1637 {
1638     //  1. HUD
1639     this->__drawHUD();
1640
1641     //  2. frame / clock overlay
1642     if (this->show_frame_clock_overlay) {
1643         this->__drawFrameClockOverlay();
1644     }
1645
1646     //  3. tutorial or turn summary
1647     if (this->show_tutorial) {
1648         this->__drawTutorial();
1649     }
1650
1651     else if (not this->turn_summary_string.empty()) {
1652         this->__drawTurnSummary();
1653     }
1654
1655     //  4. turn advance banner
1656     if (this->draw_turn_advance_banner) {
1657         this->__drawTurnAdvanceBanner();
1658     }
1659
1660     //  5. title transition
1661     if (this->transition_from_title) {
1662         this->render_window_ptr->draw(this->fade_rectangle);
1663
1664         double alpha = this->fade_rectangle.getFillColor().a;
1665
1666         alpha -= FRAMES_PER_SECOND / 20;
1667
1668         if (alpha < 0) {
1669             alpha = 0;
1670             this->transition_from_title = false;
1671         }
1672
1673         this->fade_rectangle.setFillColor(sf::Color(0, 0, 0, alpha));
1674     }
1675
1676     //  6. terminating conditions
1677     switch (this->game_phase) {
1678         case (GamePhase :: LOSS_DEMAND): {
1679             this->__drawLossDemand();
1680
1681             break;
1682         }
```

```
1683
1684
1685         case (GamePhase :: LOSS_CREDITS): {
1686             this->__drawLossCredits();
1687
1688             break;
1689         }
1690
1691
1692         case (GamePhase :: LOSS_EMISSIONS): {
1693             this->__drawLossEmissions();
1694
1695             break;
1696         }
1697
1698
1699         case (GamePhase ::VICTORY): {
1700             this->__drawVictory();
1701
1702             break;
1703         }
1704
1705
1706         default: {
1707             // do nothing!
1708
1709             break;
1710         }
1711     }
1712
1713     return;
1714 }   /* draw() */
```

4.5.3.6 __drawFrameClockOverlay()

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

Helper method to draw frame clock overlay.

```
1460 {
1461     std::string frame_clock_string = "FRAME: ";
1462     frame_clock_string += std::to_string(this->frame);
1463     frame_clock_string += "\nTIME SINCE START [s]: ";
1464     frame_clock_string += std::to_string(this->time_since_start_s);
1465
1466     sf::Text frame_clock_text(
1467         frame_clock_string,
1468         *(this->assets_manager_ptr->getFont("DroidSansMono")),
1469         16
1470     );
1471
1472     sf::RectangleShape frame_clock_backing(
1473         sf::Vector2f(
1474             1.02 * frame_clock_text.getLocalBounds().width,
1475             1.20 * frame_clock_text.getLocalBounds().height
1476         )
1477     );
1478     frame_clock_backing.setFillColor(sf::Color(0, 0, 0, 255));
1479
1480     this->render_window_ptr->draw(frame_clock_backing);
1481     this->render_window_ptr->draw(frame_clock_text);
1482
1483     return;
1484 }   /* __drawFrameClockOverlay() */
```

4.5.3.7 __drawHUD()

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

Helper method to heads-up display (HUD).

```
1499 {
1500     //  1. first line (top)
1501     std::string HUD_string = "YEAR: ";
1502     HUD_string += std::to_string(this->year);
1503 
1504     HUD_string += "    MONTH: ";
1505     HUD_string += std::to_string(this->month);
1506 
1507     HUD_string += "    POPULATION: ";
1508     HUD_string += std::to_string(this->population);
1509 
1510     HUD_string += "    CREDITS: ";
1511     HUD_string += std::to_string(this->credits);
1512     HUD_string += " K";
1513 
1514     HUD_string += "    CURRENT DEMAND: ";
1515     HUD_string += std::to_string(this->demand_MWh);
1516     HUD_string += " MWh";
1517 
1518     sf::Text HUD_text(
1519         HUD_string,
1520         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1521         16
1522     );
1523 
1524     HUD_text.setPosition(
1525         (800 - HUD_text.getLocalBounds().width) / 2,
1526         8
1527     );
1528 
1529     HUD_text.setFillColor(MONOCHROME_TEXT_GREEN);
1530 
1531     this->render_window_ptr->draw(HUD_text);
1532 
1533 
1534     //  2. second line (top)
1535     HUD_string = "CUMULATIVE EMISSIONS: ";
1536     HUD_string += std::to_string(this->cumulative_emissions_tonnes);
1537     HUD_string += " tonnes (CO2e)";
1538 
1539     HUD_string += "    LIFETIME LIMIT: ";
1540     HUD_string += std::to_string(EMISSIONS_LIFETIME_LIMIT_TONNES);
1541     HUD_string += " tonnes (CO2e)";
1542 
1543     HUD_text.setString(HUD_string);
1544 
1545     HUD_text.setPosition(
1546         (800 - HUD_text.getLocalBounds().width) / 2,
1547         35
1548     );
1549 
1550     this->render_window_ptr->draw(HUD_text);
1551 
1552 
1553     //  3. third line (bottom)
1554     HUD_string = "GAME PHASE: ";
1555 
1556     switch (this->game_phase) {
1557         case (GamePhase :: BUILD_SETTLEMENT): {
1558             HUD_string += "BUILD SETTLEMENT";
1559 
1560             break;
1561         }
1562 
1563 
1564         case (GamePhase :: SYSTEM_MANAGEMENT): {
1565             HUD_string += "SYSTEM MANAGEMENT";
1566 
1567             break;
1568         }
1569 
1570 
1571         case (GamePhase :: LOSS_EMISSIONS): {
1572             HUD_string += "LOSS (EMISSIONS)";
1573 
1574             break;
1575         }
1576 
1577 
1578         case (GamePhase :: LOSS_DEMAND): {
1579             HUD_string += "LOSS (DEMAND)";
1580 
1581             break;
1582         }
1583 
1584 
```

```
1585         case (GamePhase :: LOSS_CREDITS): {
1586             HUD_string += "LOSS (CREDITS)";
1587 
1588             break;
1589         }
1590 
1591 
1592         case (GamePhase :: VICTORY): {
1593             HUD_string += "VICTORY";
1594 
1595             break;
1596         }
1597 
1598 
1599         default: {
1600             HUD_string += "???";
1601 
1602             break;
1603         }
1604     }
1605 
1606     HUD_string += "    TURN: ";
1607     HUD_string += std::to_string(this->turn);
1608 
1609     HUD_string += "    CONSECUTIVE ZERO EMISSIONS MONTHS: ";
1610     HUD_string += std::to_string(this->consecutive_zero_emissions_months);
1611 
1612     HUD_text.setString(HUD_string);
1613 
1614     HUD_text.setPosition(
1615         (800 - HUD_text.getLocalBounds().width) / 2,
1616         GAME_HEIGHT - 35
1617     );
1618 
1619     this->render_window_ptr->draw(HUD_text);
1620 
1621     return;
1622 }   /* __drawHUD() */
```

4.5.3.8 __drawLossCredits()

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

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

```
1101 {
1102     //  1. construct loss text and backing rectangle
1103     std::string loss_credits_string  = "    LOSS! - RAN OUT OF CREDITS     \n";
1104     loss_credits_string             += "     press any key to restart      ";
1105 
1106     sf::Text loss_credits_text(
1107         loss_credits_string,
1108         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1109         32
1110     );
1111 
1112     loss_credits_text.setOrigin(
1113         loss_credits_text.getLocalBounds().width / 2,
1114         loss_credits_text.getLocalBounds().height / 2
1115     );
1116 
1117     loss_credits_text.setPosition(400, GAME_HEIGHT / 2);
1118 
1119     sf::RectangleShape backing_rectangle(
1120         sf::Vector2f(
1121             800,
1122             1.5 * loss_credits_text.getLocalBounds().height
1123         )
1124     );
1125 
1126     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1127 
1128     backing_rectangle.setOrigin(
1129         backing_rectangle.getLocalBounds().width / 2,
1130         backing_rectangle.getLocalBounds().height / 2
1131     );
1132 
1133     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
```

```
1134
1135     //  3. colour cycle and draw
1136     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1137         loss_credits_text.setFillColor(MONOCHROME_TEXT_RED);
1138     }
1139
1140     else {
1141         loss_credits_text.setFillColor(sf::Color(255, 255, 255, 255));
1142     }
1143
1144     this->render_window_ptr->draw(backing_rectangle);
1145     this->render_window_ptr->draw(loss_credits_text);
1146
1147     return;
1148 }   /* __drawLossCredits() */
```

4.5.3.9 __drawLossDemand()

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

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

```
1039 {
1040     //  1. construct alarm text and backing rectangle
1041     std::string loss_demand_string  = "   LOSS! - FAILED TO MEET DEMAND   \n";
1042     loss_demand_string             += "     press any key to restart      ";
1043
1044     sf::Text loss_demand_text(
1045         loss_demand_string,
1046         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1047         32
1048     );
1049
1050     loss_demand_text.setOrigin(
1051         loss_demand_text.getLocalBounds().width / 2,
1052         loss_demand_text.getLocalBounds().height / 2
1053     );
1054
1055     loss_demand_text.setPosition(400, GAME_HEIGHT / 2);
1056
1057     sf::RectangleShape backing_rectangle(
1058         sf::Vector2f(
1059             800,
1060             1.5 * loss_demand_text.getLocalBounds().height
1061         )
1062     );
1063
1064     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1065
1066     backing_rectangle.setOrigin(
1067         backing_rectangle.getLocalBounds().width / 2,
1068         backing_rectangle.getLocalBounds().height / 2
1069     );
1070
1071     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1072
1073     //  3. colour cycle and draw
1074     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1075         loss_demand_text.setFillColor(MONOCHROME_TEXT_RED);
1076     }
1077
1078     else {
1079         loss_demand_text.setFillColor(sf::Color(255, 255, 255, 255));
1080     }
1081
1082     this->render_window_ptr->draw(backing_rectangle);
1083     this->render_window_ptr->draw(loss_demand_text);
1084
1085     return;
1086 }   /* __drawLossDemand() */
```

4.5.3.10 __drawLossEmissions()

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

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

```
1163 {
1164     //  1. construct loss text and backing rectangle
1165     std::string loss_emissions_string  = "    LOSS! - EXCESSIVE EMISSIONS    \n";
1166     loss_emissions_string             += "     press any key to restart      ";
1167 
1168     sf::Text loss_emissions_text(
1169         loss_emissions_string,
1170         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1171         32
1172     );
1173 
1174     loss_emissions_text.setOrigin(
1175         loss_emissions_text.getLocalBounds().width / 2,
1176         loss_emissions_text.getLocalBounds().height / 2
1177     );
1178 
1179     loss_emissions_text.setPosition(400, GAME_HEIGHT / 2);
1180 
1181     sf::RectangleShape backing_rectangle(
1182         sf::Vector2f(
1183             800,
1184             1.5 * loss_emissions_text.getLocalBounds().height
1185         )
1186     );
1187 
1188     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1189 
1190     backing_rectangle.setOrigin(
1191         backing_rectangle.getLocalBounds().width / 2,
1192         backing_rectangle.getLocalBounds().height / 2
1193     );
1194 
1195     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1196 
1197     //  3. colour cycle and draw
1198     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1199         loss_emissions_text.setFillColor(MONOCHROME_TEXT_RED);
1200     }
1201 
1202     else {
1203         loss_emissions_text.setFillColor(sf::Color(255, 255, 255, 255));
1204     }
1205 
1206     this->render_window_ptr->draw(backing_rectangle);
1207     this->render_window_ptr->draw(loss_emissions_text);
1208 
1209     return;
1210 }   /* __drawLossEmissions() */
```

4.5.3.11 __drawTurnAdvanceBanner()

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

Helper method to draw turn advance banner.

```
1287 {
1288     //  1. construct advance banner text
1289     std::string turn_advance_banner_string  = "      Turn: ";
1290     turn_advance_banner_string             += std::to_string(this->turn);
1291     turn_advance_banner_string             += "\n";
1292     turn_advance_banner_string             += "Year: ";
1293     turn_advance_banner_string             += std::to_string(this->year);
1294     turn_advance_banner_string             += "    Month: ";
1295     turn_advance_banner_string             += std::to_string(this->month);
1296 
1297     sf::Text turn_advance_banner_text(
1298         turn_advance_banner_string,
1299         *(this->assets_manager_ptr->getFont("DroidSansMono")),
1300         24
```

```
1301     );
1302
1303     turn_advance_banner_text.setOrigin(
1304         turn_advance_banner_text.getLocalBounds().width / 2,
1305         turn_advance_banner_text.getLocalBounds().height / 2
1306     );
1307
1308     turn_advance_banner_text.setPosition(400, GAME_HEIGHT / 2);
1309
1310     turn_advance_banner_text.setFillColor(sf::Color(0, 0, 0, this->turn_advance_alpha));
1311
1312
1313     //  2. construct advance banner backing
1314     sf::RectangleShape backing_rectangle(
1315         sf::Vector2f(
1316             800,
1317             1.5 * turn_advance_banner_text.getLocalBounds().height
1318         )
1319     );
1320
1321     sf::Color backing_colour = RESOURCE_CHIP_GREY;
1322     backing_colour.a = this->turn_advance_alpha;
1323
1324     backing_rectangle.setFillColor(backing_colour);
1325
1326     backing_rectangle.setOrigin(
1327         backing_rectangle.getLocalBounds().width / 2,
1328         backing_rectangle.getLocalBounds().height / 2
1329     );
1330
1331     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1332
1333
1334     //  3. draw
1335     this->render_window_ptr->draw(backing_rectangle);
1336     this->render_window_ptr->draw(turn_advance_banner_text);
1337
1338     //  4. adjust alpha, check terminating conditions
1339     if (this->increase_turn_advance_alpha) {
1340         this->turn_advance_alpha += 180 * SECONDS_PER_FRAME;
1341
1342         if (this->turn_advance_alpha >= 255) {
1343             this->turn_advance_alpha = 255;
1344             this->increase_turn_advance_alpha = false;
1345         }
1346     }
1347
1348     else {
1349         this->turn_advance_alpha -= 180 * SECONDS_PER_FRAME;
1350
1351         if (this->turn_advance_alpha <= 0) {
1352             this->draw_turn_advance_banner = false;
1353         }
1354     }
1355
1356     return;
1357 }   /* __drawTurnAdvanceBanner() */
```

4.5.3.12 __drawTurnSummary()

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

Helper method to draw turn summary.

```
1416 {
1417     if (this->substring_idx < this->turn_summary_string.size()) {
1418         this->assets_manager_ptr->getSound("console string print")->play();
1419
1420         this->turn_summary_text.setString(
1421             this->turn_summary_string.substr(0, this->substring_idx)
1422         );
1423
1424         while (
1425             (this->turn_summary_string.substr(0, this->substring_idx).back() == ' ') or
1426             (this->turn_summary_string.substr(0, this->substring_idx).back() == '\n')
1427         ) {
1428             this->substring_idx++;
1429
```

```
1430             if (this->substring_idx == this->turn_summary_string.size() - 1) {
1431                 this->turn_summary_text.setString(
1432                     this->turn_summary_string.substr(0, this->substring_idx)
1433                 );
1434 
1435                 break;
1436             }
1437         }
1438 
1439         this->substring_idx++;
1440     }
1441 
1442     this->render_window_ptr->draw(this->turn_summary_text);
1443 
1444     return;
1445 }   /* __drawTurnSummary() */
```

4.5.3.13 __drawTutorial()

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

Helper method to draw tutorial text.

```
1372 {
1373     if (this->substring_idx < this->tutorial_string.size()) {
1374         this->assets_manager_ptr->getSound("console string print")->play();
1375 
1376         this->tutorial_text.setString(
1377             this->tutorial_string.substr(0, this->substring_idx)
1378         );
1379 
1380         while (
1381             (this->tutorial_string.substr(0, this->substring_idx).back() == ' ') or
1382             (this->tutorial_string.substr(0, this->substring_idx).back() == '\n')
1383         ) {
1384             this->substring_idx++;
1385 
1386             if (this->substring_idx == this->tutorial_string.size() - 1) {
1387                 this->tutorial_text.setString(
1388                     this->tutorial_string.substr(0, this->substring_idx)
1389                 );
1390 
1391                 break;
1392             }
1393         }
1394 
1395         this->substring_idx++;
1396     }
1397 
1398     this->render_window_ptr->draw(this->tutorial_text);
1399 
1400     return;
1401 }   /* __drawTutorial() */
```

4.5.3.14 __drawVictory()

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

Helper method to draw victory pop-up.

```
1225 {
1226     //  1. construct victory text and backing rectangle
1227     std::string victory_string  = "        **** VICTORY! ****         \n";
1228     victory_string             += "     press any key to restart      ";
1229 
1230     sf::Text victory_text(
1231         victory_string,
1232         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
1233         32
```

```
1234     );
1235
1236     victory_text.setOrigin(
1237         victory_text.getLocalBounds().width / 2,
1238         victory_text.getLocalBounds().height / 2
1239     );
1240
1241     victory_text.setPosition(400, GAME_HEIGHT / 2);
1242
1243     sf::RectangleShape backing_rectangle(
1244         sf::Vector2f(
1245             800,
1246             1.5 * victory_text.getLocalBounds().height
1247         )
1248     );
1249
1250     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
1251
1252     backing_rectangle.setOrigin(
1253         backing_rectangle.getLocalBounds().width / 2,
1254         backing_rectangle.getLocalBounds().height / 2
1255     );
1256
1257     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
1258
1259     //  3. colour cycle and draw
1260     if (this->frame % FRAMES_PER_SECOND <= FRAMES_PER_SECOND / 2) {
1261         victory_text.setFillColor(MONOCHROME_TEXT_GREEN);
1262     }
1263
1264     else {
1265         victory_text.setFillColor(sf::Color(255, 255, 255, 255));
1266     }
1267
1268     this->render_window_ptr->draw(backing_rectangle);
1269     this->render_window_ptr->draw(victory_text);
1270
1271     return;
1272 }    /* __drawVictory() */
```

4.5.3.15 __handleImprovementStateMessage()

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

Helper method to handle improvement state messages.

```
464 {
465     //  1. dispatch
466     if (improvement_state_message.int_payload.count("dispatch_MWh") > 0) {
467         this->demand_served_MWh += improvement_state_message.int_payload["dispatch_MWh"];
468     }
469
470     //  2. fuel costs
471     if (improvement_state_message.int_payload.count("fuel_cost") > 0) {
472         this->turn_fuel_cost += improvement_state_message.int_payload["fuel_cost"];
473     }
474
475     //  3. operation and maintenance costs
476     if (improvement_state_message.int_payload.count("operation_maintenance_cost") > 0) {
477         this->turn_operation_maintenance_cost +=
478             improvement_state_message.int_payload["operation_maintenance_cost"];
479     }
480
481     //  4. emissions
482     if (improvement_state_message.int_payload.count("emissions_tonnes_CO2e") > 0) {
483         double emissions_tonnes_CO2e =
484             improvement_state_message.int_payload["emissions_tonnes_CO2e"];
485
486         this->cumulative_emissions_tonnes += emissions_tonnes_CO2e;
487         this->turn_emissions_tonnes += emissions_tonnes_CO2e;
488     }
489
490     return;
491 }    /* __handleImprovementStateMessage() */
```

4.5.3.16 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
352 {
353     switch (this->event.key.code) {
354         case (sf::Keyboard::Enter): {
355             if (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT) {
356                 this->__advanceTurn();
357             }
358 
359             break;
360         }
361 
362 
363         case (sf::Keyboard::Tilde): {
364             this->__toggleFrameClockOverlay();
365 
366             break;
367         }
368 
369 
370         case (sf::Keyboard::Tab): {
371             this->hex_map_ptr->toggleResourceOverlay();
372 
373             break;
374         }
375 
376 
377         case (sf::Keyboard::T): {
378             this->__toggleTutorial();
379 
380             break;
381         }
382 
383 
384         case (sf::Keyboard::Left): {
385             if (this->show_tutorial) {
386                 this->__decrementTutorial();
387             }
388 
389             break;
390         }
391 
392 
393         case (sf::Keyboard::Right): {
394             if (this->show_tutorial) {
395                 this->__incrementTutorial();
396             }
397 
398             break;
399         }
400 
401 
402         default: {
403             // do nothing!
404 
405             break;
406         }
407     }
408 
409     return;
410 }   /* __handleKeyPressEvents() */
```

4.5.3.17 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
425 {
426     switch (this->event.mouseButton.button) {
427         case (sf::Mouse::Left): {
```

```
428             //...
429
430             break;
431         }
432
433
434         case (sf::Mouse::Right): {
435             //...
436
437             break;
438         }
439
440
441         default: {
442             // do nothing!
443
444             break;
445         }
446     }
447
448     return;
449 }   /* __handleMouseButtonEvents() */
```

4.5.3.18 __incrementTutorial()

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

Helper method to increment tutorial page (with wrap around).

```
292 {
293     if (this->tutorial_page == TUTORIAL_PAGES.size() - 1) {
294         this->tutorial_page = 0;
295     }
296
297     else {
298         this->tutorial_page++;
299     }
300
301     this->tutorial_string = TUTORIAL_PAGES[this->tutorial_page];
302     this->substring_idx = 0;
303
304     this->assets_manager_ptr->getSound("interface click")->play();
305
306     return;
307 }   /* __incrementTutorial() */
```

4.5.3.19 __insufficientCreditsAlarm()

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

Helper method to sound and display and insufficient credits alarm.

```
806 {
807     //  1. sound buzzer
808     this->assets_manager_ptr->getSound("insufficient credits")->play();
809
810     //  2. construct alarm text and backing rectangle
811     sf::Text insufficient_credits_text(
812         "INSUFFICIENT CREDITS",
813         (*(this->assets_manager_ptr->getFont("DroidSansMono"))),
814         32
815     );
816
817     insufficient_credits_text.setOrigin(
818         insufficient_credits_text.getLocalBounds().width / 2,
819         insufficient_credits_text.getLocalBounds().height / 2
820     );
821
822     insufficient_credits_text.setPosition(400, GAME_HEIGHT / 2);
```

```
823
824     sf::RectangleShape backing_rectangle(
825         sf::Vector2f(
826             1.1 * insufficient_credits_text.getLocalBounds().width,
827             1.5 * insufficient_credits_text.getLocalBounds().height
828         )
829     );
830
831     backing_rectangle.setFillColor(RESOURCE_CHIP_GREY);
832
833     backing_rectangle.setOrigin(
834         backing_rectangle.getLocalBounds().width / 2,
835         backing_rectangle.getLocalBounds().height / 2
836     );
837
838     backing_rectangle.setPosition(400, (GAME_HEIGHT / 2) + 8);
839
840     //  3. display loop (blocking ~3 seconds)
841     bool red_flag = true;
842     int alarm_frame = 0;
843     double time_since_alarm_s = 0;
844
845     sf::Clock alarm_clock;
846
847     while (alarm_frame < 2.5 * FRAMES_PER_SECOND) {
848
849
850         time_since_alarm_s = alarm_clock.getElapsedTime().asSeconds();
851
852         if (time_since_alarm_s >= (alarm_frame + 1) * SECONDS_PER_FRAME) {
853             while (this->render_window_ptr->pollEvent(this->event)) {
854                 // do nothing!
855             }
856
857             this->render_window_ptr->clear();
858
859             this->hex_map_ptr->draw();
860             this->context_menu_ptr->draw();
861             this->__draw();
862
863             if (alarm_frame % (FRAMES_PER_SECOND / 3) == 0) {
864                 if (red_flag) {
865                     red_flag = false;
866                 }
867
868                 else {
869                     red_flag = true;
870                 }
871             }
872
873             if (red_flag) {
874                 insufficient_credits_text.setFillColor(MONOCHROME_TEXT_RED);
875             }
876
877             else {
878                 insufficient_credits_text.setFillColor(sf::Color(255, 255, 255));
879             }
880
881             this->render_window_ptr->draw(backing_rectangle);
882             this->render_window_ptr->draw(insufficient_credits_text);
883
884             this->render_window_ptr->display();
885
886             alarm_frame++;
887             this->frame++;
888         }
889
890         // check track status, move to next if stopped
891         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
892             this->assets_manager_ptr->nextTrack();
893             this->assets_manager_ptr->playTrack();
894         }
895     }
896
897     return;
898 }   /* __insufficientCreditsAlarm( */
```

4.5.3.20 __processEvent()

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

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

```
506 {
507     if (this->event.type == sf::Event::Closed) {
508         this->quit_game = true;
509         this->game_loop_broken = true;
510     }
511
512     if (this->event.type == sf::Event::KeyPressed) {
513         this->__handleKeyPressEvents();
514     }
515
516     if (this->event.type == sf::Event::MouseButtonPressed) {
517         this->__handleMouseButtonEvents();
518     }
519
520     return;
521 }   /* __processEvent() */
```

4.5.3.21 __processMessage()

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

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

```
677 {
678     if (not this->message_hub.isEmpty(GAME_CHANNEL)) {
679         Message game_channel_message = this->message_hub.receiveMessage(GAME_CHANNEL);
680
681         if (game_channel_message.subject == "quit game") {
682             this->quit_game = true;
683             this->game_loop_broken = true;
684
685             std::cout « "Quit game message received by " « this « std::endl;
686             this->message_hub.popMessage(GAME_CHANNEL);
687         }
688
689         if (game_channel_message.subject == "restart game") {
690             this->game_loop_broken = true;
691
692             std::cout « "Restart game message received by " « this « std::endl;
693             this->message_hub.popMessage(GAME_CHANNEL);
694         }
695
696         if (game_channel_message.subject == "state request") {
697             std::cout « "Game state request message received by " « this « std::endl;
698
699             this->__sendGameStateMessage();
700             this->message_hub.popMessage(GAME_CHANNEL);
701         }
702
703         if (game_channel_message.subject == "credits spent") {
704             this->credits -= game_channel_message.int_payload["credits spent"];
705
706             std::cout « "Credits spent message (" «
707                 game_channel_message.int_payload["credits spent"] « ") received by "
708                 « this « std::endl;
709
710             std::cout « "Current credits (Game): " « this->credits « " K" «
711                 std::endl;
712
713             this->message_hub.popMessage(GAME_CHANNEL);
714         }
715
716         if (game_channel_message.subject == "insufficient credits") {
717             std::cout « "Insufficient credits message received by " « this «
718                 std::endl;
719
720             this->__insufficientCreditsAlarm();
721
722             this->message_hub.popMessage(GAME_CHANNEL);
723         }
724
725         if (game_channel_message.subject == "update game phase") {
726             std::cout « "Update game phase message received by " « this « std::endl;
727
728             if (
729                 game_channel_message.string_payload["game phase"] == "system management"
730             ) {
```

```
731                 this->game_phase = GamePhase :: SYSTEM_MANAGEMENT;
732                 this->__advanceTurn();
733             }
734 
735             else if (
736                 game_channel_message.string_payload["game phase"] == "loss emissions"
737             ) {
738                 this->game_phase = GamePhase :: LOSS_EMISSIONS;
739             }
740 
741             else if (
742                 game_channel_message.string_payload["game phase"] == "loss demand"
743             ) {
744                 this->game_phase = GamePhase :: LOSS_DEMAND;
745             }
746 
747             else if (
748                 game_channel_message.string_payload["game phase"] == "loss credits"
749             ) {
750                 this->game_phase = GamePhase :: LOSS_CREDITS;
751             }
752 
753             else if (
754                 game_channel_message.string_payload["game phase"] == "victory"
755             ) {
756                 this->game_phase = GamePhase :: VICTORY;
757             }
758 
759             this->message_hub.popMessage(GAME_CHANNEL);
760         }
761 
762         if (game_channel_message.subject == "improvement state") {
763             std::cout « "Improvement state message received by " « this « std::endl;
764 
765             this->__handleImprovementStateMessage(game_channel_message);
766 
767             this->message_hub.popMessage(GAME_CHANNEL);
768         }
769     }
770 
771     if (not this->message_hub.isEmpty(GAME_STATE_CHANNEL)) {
772         Message game_state_message =
773             this->message_hub.receiveMessage(GAME_STATE_CHANNEL);
774 
775         if (game_state_message.subject == "turn advance") {
776             if (game_state_message.number_of_reads > 0) {
777                 std::cout « "Turn advance message received by " « this « std::endl;
778                 this->message_hub.popMessage(GAME_STATE_CHANNEL);
779             }
780         }
781 
782         if (game_state_message.subject == "game state") {
783             if (game_state_message.number_of_reads > 0) {
784                 std::cout « "Game state message received by " « this « std::endl;
785                 this->message_hub.popMessage(GAME_STATE_CHANNEL);
786             }
787         }
788     }
789 
790     return;
791 }   /* __processMessage() */
```

4.5.3.22 __sendCreditsEarnedMessage()

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

Helper method to format and send a credits earned message.

```
652 {
653     Message credits_earned_message;
654 
655     credits_earned_message.channel = SETTLEMENT_CHANNEL;
656     credits_earned_message.subject = "credits earned";
657 
658     this->message_hub.sendMessage(credits_earned_message);
659 
660     std::cout « "Credits earned message sent by " « this « std::endl;
661     return;
662 }   /* __sendCreditsEarnedMessage() */
```

4.5.3.23 __sendGameStateMessage()

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

Helper method to format and send a game state message.

```
536 {
537     Message game_state_message;
538 
539     game_state_message.channel = GAME_STATE_CHANNEL;
540     game_state_message.subject = "game state";
541 
542     game_state_message.int_payload["year"] = this->year;
543     game_state_message.int_payload["month"] = this->month;
544     game_state_message.int_payload["population"] = this->population;
545     game_state_message.int_payload["credits"] = this->credits;
546     game_state_message.int_payload["demand_MWh"] = this->demand_MWh;
547     game_state_message.int_payload["cumulative_emissions_tonnes"] =
548         this->cumulative_emissions_tonnes;
549 
550     game_state_message.int_payload["reads"] = 0;
551 
552     switch (this->game_phase) {
553         case (GamePhase :: BUILD_SETTLEMENT): {
554             game_state_message.string_payload["game phase"] = "build settlement";
555 
556             break;
557         }
558 
559 
560         case (GamePhase :: SYSTEM_MANAGEMENT): {
561             game_state_message.string_payload["game phase"] = "system management";
562 
563             break;
564         }
565 
566 
567         case (GamePhase :: LOSS_EMISSIONS): {
568             game_state_message.string_payload["game phase"] = "loss emissions";
569 
570             break;
571         }
572 
573 
574         case (GamePhase :: LOSS_DEMAND): {
575             game_state_message.string_payload["game phase"] = "loss demand";
576 
577             break;
578         }
579 
580 
581         case (GamePhase :: LOSS_CREDITS): {
582             game_state_message.string_payload["game phase"] = "loss credits";
583 
584             break;
585         }
586 
587 
588         case (GamePhase :: VICTORY): {
589             game_state_message.string_payload["game phase"] = "victory";
590 
591             break;
592         }
593 
594 
595         default: {
596             // do nothing!
597 
598             break;
599         }
600     }
601 
602     game_state_message.vector_payload["demand_vec_MWh"] = this->demand_vec_MWh;
603 
604     this->message_hub.sendMessage(game_state_message);
605 
606     std::cout « "Game state message sent by " « this « std::endl;
607     return;
608 }   /* __sendGameStateMessage() */
```

4.5.3.24 __sendTurnAdvanceMessage()

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

Helper method to format and send a turn advance message.

```
623 {
624     Message turn_advance_message;
625 
626     turn_advance_message.channel = GAME_STATE_CHANNEL;
627     turn_advance_message.subject = "turn advance";
628 
629     turn_advance_message.int_payload["credits"] = this->credits;
630     turn_advance_message.int_payload["month"] = this->month;
631     turn_advance_message.int_payload["demand_MWh"] = this->demand_MWh;
632 
633     this->message_hub.sendMessage(turn_advance_message);
634 
635     std::cout << "Turn advance message sent by " << this << std::endl;
636     return;
637 }   /* __sendTurnAdvanceMessage() */
```

4.5.3.25 __summarizeTurn()

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

Helper method to generate end of turn summary.

```
913 {
914     if (this->turn - 1 == 0) {
915         return;
916     }
917 
918     this->substring_idx = 0;
919 
920     //  1. handle dispatch and demand
921     if (this->demand_served_MWh > this->past_demand_MWh) {
922         this->overproduction_MWh = this->demand_served_MWh - this->past_demand_MWh;
923         this->demand_served_MWh -= this->overproduction_MWh;
924 
925         this->overproduction_penalty =
926             round(CREDITS_PER_MWH_SERVED * this->overproduction_MWh);
927     }
928 
929     else if (this->demand_served_MWh < this->past_demand_MWh) {
930         this->demand_remaining_MWh = this->past_demand_MWh - this->demand_served_MWh;
931     }
932 
933     //  2. compute dispatch income
934     this->dispatch_income = round(CREDITS_PER_MWH_SERVED * this->demand_served_MWh);
935 
936     if (this->dispatch_income > 0) {
937         this->__sendCreditsEarnedMessage();
938     }
939 
940     //  3. compute net credit flow
941     this->net_credit_flow = this->dispatch_income -
942         this->overproduction_penalty -
943         this->turn_fuel_cost -
944         this->turn_operation_maintenance_cost;
945 
946     this->credits += this->net_credit_flow;
947 
948     //  4. assemble turn summary string
949     this->turn_summary_string.clear();
950 
951     //16 line x 32 char console  "                                \n";
952     this->turn_summary_string  = "     **** TURN ";
953     this->turn_summary_string += std::to_string(this->turn - 1);
954     this->turn_summary_string += " SUMMARY ****   \n";
955     this->turn_summary_string += "                                \n";
956 
957     this->turn_summary_string += "DEMAND:            ";
958     this->turn_summary_string += std::to_string(this->past_demand_MWh);
959     this->turn_summary_string += " MWh\n";
```

```
960
961     this->turn_summary_string += "DEMAND SERVED:     ";
962     this->turn_summary_string += std::to_string(this->demand_served_MWh);
963     this->turn_summary_string += " MWh\n";
964
965     if (this->overproduction_MWh > 0) {
966         this->turn_summary_string += "OVERPRODUCTION:    ";
967         this->turn_summary_string += std::to_string(this->overproduction_MWh);
968         this->turn_summary_string += " MWh\n";
969     }
970
971     else if (this->demand_remaining_MWh > 0) {
972         this->turn_summary_string += "DEMAND REMAINING:  ";
973         this->turn_summary_string += std::to_string(this->demand_remaining_MWh);
974         this->turn_summary_string += " MWh\n";
975     }
976
977     this->turn_summary_string += "                                  \n";
978     this->turn_summary_string += "                                  \n";
979
980     this->turn_summary_string += "DISPATCH INCOME:  +";
981     this->turn_summary_string += std::to_string(this->dispatch_income);
982     this->turn_summary_string += " K\n";
983
984     this->turn_summary_string += "FUEL COST:        -";
985     this->turn_summary_string += std::to_string(this->turn_fuel_cost);
986     this->turn_summary_string += " K\n";
987
988     this->turn_summary_string += "OP & MAINT COST:  -";
989     this->turn_summary_string += std::to_string(this->turn_operation_maintenance_cost);
990     this->turn_summary_string += " K\n";
991
992     this->turn_summary_string += "OVERPRODUCTION:   -";
993     this->turn_summary_string += std::to_string(this->overproduction_penalty);
994     this->turn_summary_string += " K\n";
995
996     this->turn_summary_string += "--------------------------------\n";
997
998     this->turn_summary_string += "NET:              ";
999
1000     if (this->net_credit_flow > 0) {
1001         this->turn_summary_string += "+";
1002     }
1003
1004     this->turn_summary_string += std::to_string(this->net_credit_flow);
1005     this->turn_summary_string += " K\n";
1006
1007     this->turn_summary_string += "                                   \n";
1008
1009     this->turn_summary_string += "EMISSIONS: ";
1010     this->turn_summary_string += std::to_string(this->turn_emissions_tonnes);
1011     this->turn_summary_string += " tonnes CO2e\n";
1012
1013     if (this->turn_emissions_tonnes <= 0) {
1014         this->consecutive_zero_emissions_months++;
1015     }
1016
1017     else {
1018         this->consecutive_zero_emissions_months = 0;
1019     }
1020
1021     //  5. send game state message
1022     this->__sendGameStateMessage();
1023
1024     return;
1025 }   /* _summarizeTurn() */
```

4.5.3.26 __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() */
```

4.5.3.27 __toggleTutorial()

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

Helper method to handle toggling the tutorial on and off.

```
263 {
264     if (this->show_tutorial) {
265         this->show_tutorial = false;
266     }
267 
268     else {
269         this->show_tutorial = true;
270     }
271 
272     this->substring_idx = 0;
273 
274     this->assets_manager_ptr->getSound("interface click")->play();
275 
276     return;
277 }   /* __toggleTutorial() */
```

4.5.3.28 __updatePopulation()

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

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

```
141 {
142     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
143     std::default_random_engine generator(seed);
144 
145     std::normal_distribution<double> normal_dist(
146         MEAN_POPULATION_GROWTH_RATE,
147         STDEV_POPULATION_GROWTH_RATE
148     );
149 
150     double growth_rate = normal_dist(generator);
151 
152     this->population = ceil((1 + growth_rate) * this->population);
153 
154     return;
155 }   /* __updatePopulation() */
```

4.5.3.29 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).

```
1869 {
1870     // start game loop
1871     while (not this->game_loop_broken) {
1872         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
1873 
1874         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
1875             //  process events
1876             while (
1877                 (not this->transition_from_title) and
1878                 (this->render_window_ptr->pollEvent(this->event))
1879             ) {
1880                 if (
1881                     (this->game_phase == GamePhase :: BUILD_SETTLEMENT) or
1882                     (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT)
1883                 ) {
1884                     this->hex_map_ptr->processEvent();
1885                     this->context_menu_ptr->processEvent();
1886                     this->__processEvent();
1887                 }
1888 
1889                 else {
1890                     if (this->event.type == sf::Event::KeyPressed) {
1891                         this->game_loop_broken = true;
1892                     }
1893                 }
1894             }
1895 
1896 
1897             //  process messages
1898             while (this->message_hub.hasTraffic()) {
1899                 this->hex_map_ptr->processMessage();
1900                 this->context_menu_ptr->processMessage();
1901                 this->__processMessage();
1902 
1903                 this->check_terminating_conditions = true;
1904 
1905                 if (not this->message_deadlock) {
1906                     this->message_deadlock_frame++;
1907 
1908                     if (this->message_deadlock_frame > 5 * FRAMES_PER_SECOND) {
1909                         this->message_hub.printState();
1910                         this->message_deadlock = true;
1911                     }
1912                 }
1913             }
1914             this->message_deadlock = false;
1915             this->message_deadlock_frame = 0;
1916 
1917 
1918             //  handle turn end summary
1919             if (this->turn_end) {
1920                 std::cout « "**** END OF TURN " « std::to_string(this->turn - 1) «
1921                     " ****" « std::endl;
1922 
1923                 this->__summarizeTurn();
1924 
1925                 this->turn_end = false;
1926 
1927                 this->draw_turn_advance_banner = true;
1928                 this->turn_advance_alpha = 0;
1929                 this->increase_turn_advance_alpha = true;
1930             }
1931 
1932 
1933             //  check terminating conditions
1934             if (this->check_terminating_conditions) {
1935                 this->__checkTerminatingConditions();
1936                 this->check_terminating_conditions = false;
1937             }
1938 
1939 
1940             //  draw frame
1941             this->render_window_ptr->clear();
1942 
1943             this->hex_map_ptr->draw();
1944             this->context_menu_ptr->draw();
1945             this->__draw();
1946 
1947             this->render_window_ptr->display();
1948 
1949 
1950             //  increment frame
1951             this->frame++;
1952         }
```

```
1953
1954         // check track status, move to next if stopped
1955         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
1956             this->assets_manager_ptr->nextTrack();
1957             this->assets_manager_ptr->playTrack();
1958         }
1959
1960     }
1961
1962     return this->quit_game;
1963 }   /* 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 draw_turn_advance_banner

```
bool Game::draw_turn_advance_banner
```

A boolean indicating whether or not to draw the turn advance banner.

4.5.4.14 event

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

The game events class.

4.5.4.15 fade_rectangle

```
sf::RectangleShape Game::fade_rectangle
```

A fading rectangle (for smooth transition from title to game).

4.5.4.16 frame

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

The current frame of the game.

4.5.4.17 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.18 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.19 hex_map_ptr

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

Pointer to the hex map (defines game world).

4.5.4.20 increase_turn_advance_alpha

```
bool Game::increase_turn_advance_alpha
```

A boolean which indicates whether the turn advance alpha is increasing or decreasing.

4.5.4.21 message_deadlock

```
bool Game::message_deadlock
```

A boolean indicating whether a message deadlock has been detected.

4.5.4.22 message_deadlock_frame

```
int Game::message_deadlock_frame
```

A frame counter for detecting message deadlock.

4.5.4.23 message_hub

```
MessageHub Game::message_hub
```

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

4.5.4.24 month

```
int Game::month
```

Current game month.

4.5.4.25 net_credit_flow

```
int Game::net_credit_flow
```

The net credit flow at the end of a turn.

4.5.4.26 overproduction_MWh

```
int Game::overproduction_MWh
```

The amount of overproduction at the end of a turn.

4.5.4.27 overproduction_penalty

```
int Game::overproduction_penalty
```

The penalty for overproduction.

4.5.4.28 past_demand_MWh

```
int Game::past_demand_MWh
```

The demand in the previous turn.

4.5.4.29 population

```
int Game::population
```

Current population.

4.5.4.30 quit_game

```
bool Game::quit_game
```

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

4.5.4.31 render_window_ptr

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

A pointer to the render window.

4.5.4.32 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

4.5.4.33 show_tutorial

```
bool Game::show_tutorial
```

A boolean indicating whether or not to show the tutorial.

4.5.4.34 substring_idx

```
size_t Game::substring_idx
```

The index of the turn summary or tutorial substring.

4.5.4.35 time_since_start_s

```
double Game::time_since_start_s
```

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

4.5.4.36 transition_from_title

```
bool Game::transition_from_title
```

A boolean which indicates if construction follows a title transition.

4.5.4.37 turn

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

The current game turn.

4.5.4.38 turn_advance_alpha

```
double Game::turn_advance_alpha
```

The alpha value for the turn advance banner.

4.5.4.39 turn_emissions_tonnes

```
int Game::turn_emissions_tonnes
```

The amount of emissions at the end of a turn.

4.5.4.40 turn_end

```
bool Game::turn_end
```

A boolean indicating a turn end.

4.5.4.41 turn_fuel_cost

```
int Game::turn_fuel_cost
```

The cost of fuel at the end of a turn.

4.5.4.42 turn_operation_maintenance_cost

```
int Game::turn_operation_maintenance_cost
```

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

4.5.4.43 turn_summary_string

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

A string representation of the end of turn summary.

4.5.4.44 turn_summary_text

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

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

4.5.4.45 tutorial_page

```
size_t Game::tutorial_page
```

Index for which page of the tutorial to show.

4.5.4.46 tutorial_string

```
std::string Game::tutorial_string
```

A string representation of the current tutorial page.

4.5.4.47 tutorial_text

`sf::Text Game::tutorial_text`

A text representation (drawable) of the tutorial page.

4.5.4.48 year

`int Game::year`

Current game year.

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

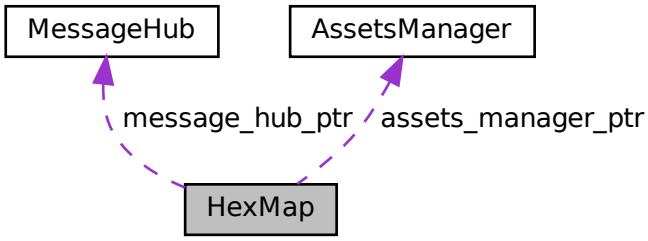
- header/Game.h
- source/Game.cpp

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.
- bool just_constructed

  A boolean which indicates if the HexMap has just been constructed.
- 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.
- size_t initial_draw_tile_idx

  The current tile idx (for the initial draw tile wave animation).
- int demand_MWh

  Current energy demand [MWh].
- double dalpha

  The change in tile alpha (for the tile wave animation).
- 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.

- void __setUpInitialDraw (void)

  Helper method to set up initial map draw (scale all tiles to zero, to support tile wave animation).

- void __handleInitialDraw (void)

  Helper method to handle initial map draw (tile wave animation).

- 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.

```
1411 {
1412     //  1. set attributes
1413
1414     //  1.1. private
1415     this->event_ptr = event_ptr;
1416     this->render_window_ptr = render_window_ptr;
1417
1418     this->assets_manager_ptr = assets_manager_ptr;
1419     this->message_hub_ptr = message_hub_ptr;
1420
1421     //  1.2. public
1422     this->show_resource = false;
1423     this->tile_selected = false;
1424     this->settlement_position_logged = false;
1425     this->just_constructed = true;
1426
1427     this->frame = 0;
1428     this->initial_draw_tile_idx = 1;
1429
1430     this->n_layers = n_layers;
1431     if (this->n_layers < 0) {
1432         this->n_layers = 0;
1433     }
1434
1435     this->demand_MWh = 0;
1436
1437     this->dalpha = 1.6 * FRAMES_PER_SECOND;
1438
1439     this->position_x = 400;
1440     this->position_y = 400;
1441
1442     this->settlement_position_x = 0;
1443     this->settlement_position_y = 0;
1444
1445     //  2. assemble n layer hex map
1446     this->__assembleHexMap();
1447
1448     //  3. set up and position drawable attributes
1449     this->__setUpGlassScreen();
1450
1451     //  4. add message channel(s)
1452     this->message_hub_ptr->addChannel(TILE_SELECTED_CHANNEL);
1453     this->message_hub_ptr->addChannel(NO_TILE_SELECTED_CHANNEL);
1454     this->message_hub_ptr->addChannel(TILE_STATE_CHANNEL);
1455     this->message_hub_ptr->addChannel(HEX_MAP_CHANNEL);
1456
1457     std::cout « "HexMap constructed at " « this « std::endl;
1458
1459     return;
1460 }   /* HexMap(), intended */
```

4.6.2.2 ~HexMap()

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

Destructor for the HexMap class.

```
1792 {
1793     this->clear();
1794
1795     std::cout « "HexMap at " « this « " destroyed" « std::endl;
1796
1797     return;
1798 }   /* ~HexMap() */
```

4.6.3 Member Function Documentation

4.6.3.1 __assembleHexMap()

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

Helper method to assemble the hex map.

```
966 {
967     //  1. seed RNG (using milliseconds since 1 Jan 1970)
968     unsigned long long int milliseconds_since_epoch =
969         std::chrono::duration_cast<std::chrono::milliseconds>(
970             std::chrono::system_clock::now().time_since_epoch()
971         ).count();
972     srand(milliseconds_since_epoch);
973 
974     //  2. lay tiles
975     this->__layTiles();
976     this->__buildDrawOrderVector();
977 
978     //  3. procedurally generate types
979     this->__procedurallyGenerateTileTypes();
980 
981     //  4. procedurally generate resources
982     this->__procedurallyGenerateTileResources();
983 
984     //  5. set up initial draw
985     this->__setUpInitialDraw();
986 
987     return;
988 }   /* __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.

```
1217 {
1218     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
1219 
1220     for (size_t i = 0; i < neighbours_vec.size(); i++) {
1221         neighbours_vec[i]->assess();
1222     }
1223 
1224     return;
1225 }   /* __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.

```
1241 {
1242     //  1. compute total production / dispatch
1243     int total_production_dispatch_MWh = 0;
1244 
1245     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1246     std::map<double, HexTile*>::iterator hex_map_iter_y;
1247 
1248     TileImprovement* tile_improvement_ptr;
1249 
1250     for (
1251         hex_map_iter_x = this->hex_map.begin();
1252         hex_map_iter_x != this->hex_map.end();
1253         hex_map_iter_x++
1254     ) {
1255         for (
1256             hex_map_iter_y = hex_map_iter_x->second.begin();
1257             hex_map_iter_y != hex_map_iter_x->second.end();
1258             hex_map_iter_y++
1259         ) {
1260             if (
1261                 (hex_map_iter_y->second->has_improvement) and
1262                 (hex_map_iter_y->second->tile_improvement_ptr->tile_improvement_type !=
```

```
1263                    TileImprovementType :: SETTLEMENT)
1264            ) {
1265                tile_improvement_ptr = hex_map_iter_y->second->tile_improvement_ptr;
1266
1267                switch (tile_improvement_ptr->tile_improvement_type) {
1268                    case (TileImprovementType :: DIESEL_GENERATOR): {
1269                        total_production_dispatch_MWh +=
1270                            ((DieselGenerator*)tile_improvement_ptr)->production_MWh;
1271
1272                        break;
1273                    }
1274
1275
1276                    case (TileImprovementType :: SOLAR_PV): {
1277                        total_production_dispatch_MWh +=
1278                            ((SolarPV*)tile_improvement_ptr)->dispatch_MWh;
1279
1280                        break;
1281                    }
1282
1283
1284                    case (TileImprovementType :: TIDAL_TURBINE): {
1285                        total_production_dispatch_MWh +=
1286                            ((TidalTurbine*)tile_improvement_ptr)->dispatch_MWh;
1287
1288                        break;
1289                    }
1290
1291
1292                    case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
1293                        total_production_dispatch_MWh +=
1294                            ((WaveEnergyConverter*)tile_improvement_ptr)->dispatch_MWh;
1295
1296                        break;
1297                    }
1298
1299
1300                    case (TileImprovementType :: WIND_TURBINE): {
1301                        total_production_dispatch_MWh +=
1302                            ((WindTurbine*)tile_improvement_ptr)->dispatch_MWh;
1303
1304                        break;
1305                    }
1306
1307
1308                    default: {
1309                        // do nothing!
1310
1311                        break;
1312                    }
1313                }
1314            }
1315        }
1316    }
1317
1318    //  2. construct total text
1319    sf::Text total_production_dispatch_text(
1320        std::to_string(total_production_dispatch_MWh),
1321        *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1322        16
1323    );
1324
1325    total_production_dispatch_text.setOrigin(
1326        total_production_dispatch_text.getLocalBounds().width / 2,
1327        total_production_dispatch_text.getLocalBounds().height / 2
1328    );
1329
1330    total_production_dispatch_text.setPosition(800 - 20, 20 - 4);
1331
1332    sf::Color text_colour;
1333
1334    if (total_production_dispatch_MWh < this->demand_MWh) {
1335        text_colour = MONOCHROME_TEXT_RED;
1336    }
1337
1338    else if (total_production_dispatch_MWh > this->demand_MWh) {
1339        text_colour = MONOCHROME_TEXT_AMBER;
1340    }
1341
1342    else {
1343        text_colour = MONOCHROME_TEXT_GREEN;
1344    }
1345
1346    total_production_dispatch_text.setFillColor(text_colour);
1347
1348    //  4. construct total backing
1349    sf::RectangleShape total_production_dispatch_backing(sf::Vector2f(32, 32));
```

```
1350
1351     total_production_dispatch_backing.setOrigin(
1352         total_production_dispatch_backing.getLocalBounds().width / 2,
1353         total_production_dispatch_backing.getLocalBounds().height / 2
1354     );
1355
1356     total_production_dispatch_backing.setPosition(800 - 20, 20);
1357
1358     total_production_dispatch_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
1359
1360     total_production_dispatch_backing.setOutlineColor(MENU_FRAME_GREY);
1361     total_production_dispatch_backing.setOutlineThickness(2);
1362
1363     //  4. draw
1364     if (total_production_dispatch_MWh > 0) {
1365         this->render_window_ptr->draw(total_production_dispatch_backing);
1366         this->render_window_ptr->draw(total_production_dispatch_text);
1367     }
1368
1369     return;
1370 }   /* __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.

```
877 {
878     std::cout « "enforcing ocean continuity ..." « std::endl;
879
880     bool tile_changed = false;
881
882     //  1. scan tiles and enforce (where appropriate)
883     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
884     std::map<double, HexTile*>::iterator hex_map_iter_y;
885     HexTile* hex_ptr;
886     for (
887         hex_map_iter_x = this->hex_map.begin();
888         hex_map_iter_x != this->hex_map.end();
889         hex_map_iter_x++
890     ) {
891         for (
892             hex_map_iter_y = hex_map_iter_x->second.begin();
893             hex_map_iter_y != hex_map_iter_x->second.end();
894             hex_map_iter_y++
895         ) {
896             hex_ptr = hex_map_iter_y->second;
897
898             if (this->__isLakeTouchingOcean(hex_ptr)) {
899                 hex_ptr->setTileType(TileType :: OCEAN);
900                 tile_changed = true;
901             }
902         }
903     }
904
905     if (tile_changed) {
906         this->__enforceOceanContinuity();
907     }
908     else {
909         return;
910     }
911 }   /* __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.

```
733 {
734     //  1. init type count map
735     std::map<TileType, int> type_count_map;
736     type_count_map[hex_ptr->tile_type] = 1;
737 
738     //  2. survey neighbours, count type instances
739     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
740 
741     for (size_t i = 0; i < neighbours_vec.size(); i++) {
742         if (type_count_map.count(neighbours_vec[i]->tile_type) <= 0) {
743             type_count_map[neighbours_vec[i]->tile_type] = 1;
744         }
745         else {
746             type_count_map[neighbours_vec[i]->tile_type] += 1;
747         }
748     }
749 
750     //  3. find majority tile type
751     int max_count = -1 * std::numeric_limits<int>::infinity();
752     TileType majority_tile_type = hex_ptr->tile_type;
753 
754     std::map<TileType, int>::iterator map_iter;
755     for (
756         map_iter = type_count_map.begin();
757         map_iter != type_count_map.end();
758         map_iter++
759     ){
760         if (map_iter->second > max_count) {
761             max_count = map_iter->second;
762             majority_tile_type = map_iter->first;
763         }
764     }
765 
766     //  4. detect ties
767     for (
768         map_iter = type_count_map.begin();
769         map_iter != type_count_map.end();
770         map_iter++
771     ){
772         if (
773             map_iter->second == max_count and
774             map_iter->first != majority_tile_type
775         ) {
776             majority_tile_type = hex_ptr->tile_type;
777             break;
778         }
779     }
780 
781     return majority_tile_type;
782 }   /* __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.

```
675 {
676     std::vector<HexTile*> neighbours_vec;
677 
678     //  1. build potential neighbour positions
679     std::vector<double> potential_neighbour_x_vec(6, 0);
680     std::vector<double> potential_neighbour_y_vec(6, 0);
681 
682     for (int i = 0; i < 6; i++) {
683         potential_neighbour_x_vec[i] = hex_ptr->position_x +
684             2 * hex_ptr->minor_radius * cos((60 * i) * (M_PI / 180));
685 
686         potential_neighbour_y_vec[i] = hex_ptr->position_y +
687             2 * hex_ptr->minor_radius * sin((60 * i) * (M_PI / 180));
688     }
689 
690     //  2. populate neighbours vector
691     std::vector<double> map_index_positions;
692     double potential_x = 0;
693     double potential_y = 0;
694 
695     for (int i = 0; i < 6; i++) {
696         potential_x = potential_neighbour_x_vec[i];
697         potential_y = potential_neighbour_y_vec[i];
698 
699         map_index_positions = this->__getValidMapIndexPositions(
700             potential_x,
701             potential_y
702         );
703 
704         if (not (map_index_positions[0] == -1)) {
705             neighbours_vec.push_back(
706                 this->hex_map[map_index_positions[0]][map_index_positions[1]]
707             );
708         }
709     }
710 
711     return neighbours_vec;
712 }   /* __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].

```
440 {
441     //  1. generate random amplitude, wave number, direction, and phase vectors
442     std::vector<double> random_amplitude_vec(n_components, 0);
443     std::vector<double> random_wave_number_vec(n_components, 0);
444     std::vector<double> random_frequency_vec(n_components, 0);
445     std::vector<double> random_direction_vec(n_components, 0);
446     std::vector<double> random_phase_vec(n_components, 0);
447 
448     for (int i = 0; i < n_components; i++) {
```

```
449         random_amplitude_vec[i] = 10 * ((double)rand() / RAND_MAX);
450
451         random_wave_number_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
452
453         random_frequency_vec[i] = ((double)rand() / RAND_MAX);
454
455         random_direction_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
456
457         random_phase_vec[i] = 2 * M_PI * ((double)rand() / RAND_MAX);
458     }
459
460     //  2. generate noise vec
461     double amp = 0;
462     double wave_no = 0;
463     double freq = 0;
464     double dir = 0;
465     double phase = 0;
466
467     double x = 0;
468     double y = 0;
469     double t = time(NULL);
470
471     double max_noise = -1 * std::numeric_limits<double>::infinity();
472     double min_noise = std::numeric_limits<double>::infinity();
473
474     double noise = 0;
475     std::vector<double> noise_vec(n_elements, 0);
476
477     for (int i = 0; i < n_elements; i++) {
478         x = this->tile_position_x_vec[i] - this->position_x;
479         y = this->tile_position_y_vec[i] - this->position_y;
480
481         for (int j = 0; j < n_components; j++) {
482             amp = random_amplitude_vec[j];
483             wave_no = random_wave_number_vec[j];
484             freq = random_frequency_vec[j];
485             dir = random_direction_vec[j];
486             phase = random_phase_vec[j];
487
488             noise += (amp / (j + 1)) * cos(
489                 wave_no * (j + 1) * (x * sin(dir) + y * cos(dir)) +
490                 2 * M_PI * (j + 1) * freq * t +
491                 phase
492             );
493         }
494
495         noise_vec[i] = noise;
496
497         if (noise > max_noise) {
498             max_noise = noise;
499         }
500
501         else if (noise < min_noise) {
502             min_noise = noise;
503         }
504
505         noise = 0;
506     }
507
508     //  3. normalize noise vec
509     for (int i = 0; i < n_elements; i++) {
510         noise_vec[i] = (noise_vec[i] - min_noise) / (max_noise - min_noise);
511
512         if (noise_vec[i] < 0) {
513             noise_vec[i] = 0;
514         }
515         else if (noise_vec[i] > 1) {
516             noise_vec[i] = 1;
517         }
518     }
519
520     return noise_vec;
521 }   /* __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).

```
1005 {
1006     HexTile* selected_tile_ptr = NULL;
1007 
1008     bool break_flag = false;
1009     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1010     std::map<double, HexTile*>::iterator hex_map_iter_y;
1011 
1012     for (
1013         hex_map_iter_x = this->hex_map.begin();
1014         hex_map_iter_x != this->hex_map.end();
1015         hex_map_iter_x++
1016     ) {
1017         for (
1018             hex_map_iter_y = hex_map_iter_x->second.begin();
1019             hex_map_iter_y != hex_map_iter_x->second.end();
1020             hex_map_iter_y++
1021         ) {
1022             if (hex_map_iter_y->second->is_selected) {
1023                 selected_tile_ptr = hex_map_iter_y->second;
1024                 break_flag = true;
1025             }
1026 
1027             if (break_flag) {
1028                 break;
1029             }
1030         }
1031 
1032         if (break_flag) {
1033             break;
1034         }
1035     }
1036 
1037     return selected_tile_ptr;
1038 }   /* __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.

```
621 {
622     std::vector<double> map_index_positions = {-1, -1};
623 
624     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
625     std::map<double, HexTile*>::iterator hex_map_iter_y;
626     HexTile* hex_ptr;
627 
628     double distance = 0;
629 
630     for (
631         hex_map_iter_x = this->hex_map.begin();
```

```
632         hex_map_iter_x != this->hex_map.end();
633         hex_map_iter_x++
634     ) {
635         for (
636             hex_map_iter_y = hex_map_iter_x->second.begin();
637             hex_map_iter_y != hex_map_iter_x->second.end();
638             hex_map_iter_y++
639         ) {
640             hex_ptr = hex_map_iter_y->second;
641 
642             distance = sqrt(
643                 pow(hex_ptr->position_x - potential_x, 2) +
644                 pow(hex_ptr->position_y - potential_y, 2)
645             );
646 
647             if (distance <= hex_ptr->minor_radius / 4) {
648                 map_index_positions = {hex_ptr->position_x, hex_ptr->position_y};
649                 return map_index_positions;
650             }
651         }
652     }
653 
654     return map_index_positions;
655 }   /* __isInHexMap() */
```

4.6.3.11 __handleInitialDraw()

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

Helper method to handle initial map draw (tile wave animation).

```
373 {
374     double alpha = 0;
375     sf::Color tile_colour(255, 255, 255, 255);
376 
377     for (size_t i = 0; i < this->initial_draw_tile_idx; i++) {
378         tile_colour = this->hex_draw_order_vec[i]->tile_sprite.getFillColor();
379         alpha = tile_colour.a;
380 
381         alpha += this->dalpha;
382 
383         if (alpha >= 255) {
384             alpha = 255;
385         }
386 
387         tile_colour.a = alpha;
388 
389         this->hex_draw_order_vec[i]->tile_sprite.setFillColor(tile_colour);
390         this->hex_draw_order_vec[i]->tile_decoration_sprite.setColor(
391             sf::Color(255, 255, 255, alpha)
392         );
393 
394         if (i < this->hex_draw_order_vec.size() - 1) {
395             if (i == this->initial_draw_tile_idx - 1) {
396                 if (alpha >= 128) {
397                     this->initial_draw_tile_idx++;
398 
399                     if (
400                         this->assets_manager_ptr->getSound("card flick")->getStatus() !=
401                         sf::SoundSource::Playing
402                     ) {
403                         this->assets_manager_ptr->getSound("card flick")->play();
404                     }
405                 }
406             }
407         }
408 
409         else {
410             if (alpha >= 255) {
411                 this->just_constructed = false;
412             }
413         }
414     }
415 
416     return;
417 }   /* __handleInitialDraw() */
```

4.6.3.12 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
1109 {
1110     switch (this->event_ptr->key.code) {
1111         case (sf::Keyboard::Escape): {
1112             this->tile_selected = false;
1113         }
1114 
1115 
1116         default: {
1117             // do nothing!
1118 
1119             break;
1120         }
1121     }
1122 
1123     return;
1124 }   /* __handleKeyPressEvents() */
```

4.6.3.13 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
1139 {
1140     switch (this->event_ptr->mouseButton.button) {
1141         case (sf::Mouse::Left): {
1142             HexTile* hex_ptr = this->__getSelectedTile();
1143 
1144             if (hex_ptr != NULL) {
1145                 this->tile_selected = true;
1146             }
1147 
1148             else if (this->tile_selected) {
1149                 this->tile_selected = false;
1150                 this->__sendNoTileSelectedMessage();
1151             }
1152 
1153             break;
1154         }
1155 
1156 
1157         case (sf::Mouse::Right): {
1158             if (this->tile_selected) {
1159                 this->tile_selected = false;
1160                 this->__sendNoTileSelectedMessage();
1161             }
1162 
1163             break;
1164         }
1165 
1166 
1167         default: {
1168             // do nothing!
1169 
1170             break;
1171         }
1172     }
1173 
1174     return;
1175 }   /* __handleMouseButtonEvents() */
```

4.6.3.14 __isLakeTouchingOcean()

```
bool HexMap::__isLakeTouchingOcean (
            HexTile * hex_ptr )  [private]
```

```
844 {
845     //  1. if not lake tile, return
846     if (not (hex_ptr->tile_type == TileType :: LAKE)) {
847         return false;
848     }
849 
850     //  2. scan neighbours for ocean tiles
851     std::vector<HexTile*> neighbours_vec = this->__getNeighboursVector(hex_ptr);
852 
853     for (size_t i = 0; i < neighbours_vec.size(); i++) {
854         if (neighbours_vec[i]->tile_type == TileType :: OCEAN) {
855             return true;
856         }
857     }
858 
859     return false;
860 }   /* __isLakeTouchingOcean() */
```

4.6.3.15 __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() */
```

4.6.3.16 __logSettlementPosition()

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

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

```
1053 {
1054     bool break_flag = false;
1055
1056     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1057     std::map<double, HexTile*>::iterator hex_map_iter_y;
1058
1059     for (
1060         hex_map_iter_x = this->hex_map.begin();
1061         hex_map_iter_x != this->hex_map.end();
1062         hex_map_iter_x++
1063     ) {
1064         for (
1065             hex_map_iter_y = hex_map_iter_x->second.begin();
1066             hex_map_iter_y != hex_map_iter_x->second.end();
1067             hex_map_iter_y++
1068         ) {
1069             if (
1070                 (hex_map_iter_y->second->has_improvement) and
1071                 (hex_map_iter_y->second->tile_improvement_ptr->tile_improvement_type ==
1072                     TileImprovementType :: SETTLEMENT)
1073             ) {
1074                 this->settlement_position_x = hex_map_iter_y->second->position_x;
1075                 this->settlement_position_y = hex_map_iter_y->second->position_y;
1076
1077                 this->settlement_position_logged = true;
1078
1079                 std::cout « "Settlement position logged, (" «
1080                     this->settlement_position_x « ", " «
1081                     this->settlement_position_y « ")" « std::endl;
1082
1083                 break_flag = true;
1084                 break;
1085             }
1086         }
1087
```

```
1088         if (break_flag) {
1089             break;
1090         }
1091     }
1092
1093     return;
1094 }   /* __logSettlementPosition() */
```

4.6.3.17 __procedurallyGenerateTileResources()

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

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

```
926 {
927     //  1. get random cosine series noise vec
928     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
929
930     //  2. set tile resources based on random cosine series noise
931     int noise_idx = 0;
932
933     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
934     std::map<double, HexTile*>::iterator hex_map_iter_y;
935     for (
936         hex_map_iter_x = this->hex_map.begin();
937         hex_map_iter_x != this->hex_map.end();
938         hex_map_iter_x++
939     ) {
940         for (
941             hex_map_iter_y = hex_map_iter_x->second.begin();
942             hex_map_iter_y != hex_map_iter_x->second.end();
943             hex_map_iter_y++
944         ) {
945             hex_map_iter_y->second->setTileResource(noise_vec[noise_idx]);
946             noise_idx++;
947         }
948     }
949
950     return;
951 }   /* __procedurallyGenerateTileResources() */
```

4.6.3.18 __procedurallyGenerateTileTypes()

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

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

```
536 {
537     //  1. get random cosine series noise vec
538     std::vector<double> noise_vec = this->__getNoise(this->n_tiles);
539
540     //  2. set initial tile types based on either random cosine series noise or white
541     //     noise (decided by coin toss)
542     int noise_idx = 0;
543
544     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
545     std::map<double, HexTile*>::iterator hex_map_iter_y;
546     for (
547         hex_map_iter_x = this->hex_map.begin();
548         hex_map_iter_x != this->hex_map.end();
549         hex_map_iter_x++
550     ) {
551         for (
552             hex_map_iter_y = hex_map_iter_x->second.begin();
553             hex_map_iter_y != hex_map_iter_x->second.end();
554             hex_map_iter_y++
555         ) {
556             if ((double)rand() / RAND_MAX > 0.5) {
557                 hex_map_iter_y->second->setTileType(noise_vec[noise_idx]);
```

```
558             }
559             else {
560                 hex_map_iter_y->second->setTileType((double)rand() / RAND_MAX);
561             }
562             noise_idx++;
563         }
564     }
565
566     //  3. smooth tile types (majority rules)
567     this->__smoothTileTypes();
568
569     //  4. set border tile type to ocean
570     for (size_t i = 0; i < this->border_tiles_vec.size(); i++) {
571         this->border_tiles_vec[i]->setTileType(TileType :: OCEAN);
572     }
573
574     //  5. enforce ocean continuity (i.e. all lake tiles touching ocean become ocean)
575     this->__enforceOceanContinuity();
576
577     //  6. decorate tiles
578     for (
579         hex_map_iter_x = this->hex_map.begin();
580         hex_map_iter_x != this->hex_map.end();
581         hex_map_iter_x++
582     ) {
583         for (
584             hex_map_iter_y = hex_map_iter_x->second.begin();
585             hex_map_iter_y != hex_map_iter_x->second.end();
586             hex_map_iter_y++
587         ) {
588             hex_map_iter_y->second->decorateTile();
589         }
590     }
591
592     return;
593 }   /* __procedurallyGenerateTileTypes() */
```

4.6.3.19 __sendNoTileSelectedMessage()

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

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

```
1190 {
1191     Message no_tile_selected_message;
1192
1193     no_tile_selected_message.channel = NO_TILE_SELECTED_CHANNEL;
1194     no_tile_selected_message.subject = "no tile selected";
1195
1196     this->message_hub_ptr->sendMessage(no_tile_selected_message);
1197
1198     std::cout « "No tile selected message sent by " « this « std::endl;
1199     return;
1200 }   /* __sendNoTileSelectedMessage() */
```

4.6.3.20 __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() */
```

4.6.3.21 __setUpInitialDraw()

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

Helper method to set up initial map draw (scale all tiles to zero, to support tile wave animation).

```
342 {
343     double alpha = 0;
344     sf::Color tile_colour(255, 255, 255, 255);
345 
346     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
347         tile_colour = this->hex_draw_order_vec[i]->tile_sprite.getFillColor();
348         tile_colour.a = alpha;
349 
350         this->hex_draw_order_vec[i]->tile_sprite.setFillColor(tile_colour);
351 
352         this->hex_draw_order_vec[i]->tile_decoration_sprite.setColor(
353             sf::Color(255, 255, 255, 0)
354         );
355     }
356 
357     return;
358 }   /* __setUpInitialDraw() */
```

4.6.3.22 __smoothTileTypes()

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

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

```
797 {
798     std::cout « "smoothing ..." « std::endl;
799 
800     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
801     std::map<double, HexTile*>::iterator hex_map_iter_y;
802     HexTile* hex_ptr;
803     TileType majority_tile_type;
804 
805     for (
806         hex_map_iter_x = this->hex_map.begin();
807         hex_map_iter_x != this->hex_map.end();
808         hex_map_iter_x++
809     ) {
810         for (
811             hex_map_iter_y = hex_map_iter_x->second.begin();
812             hex_map_iter_y != hex_map_iter_x->second.end();
813             hex_map_iter_y++
814         ) {
815             hex_ptr = hex_map_iter_y->second;
816             majority_tile_type = this->__getMajorityTileType(hex_ptr);
817 
818             if (majority_tile_type != hex_ptr->tile_type) {
819                 hex_ptr->setTileType(majority_tile_type);
820             }
821         }
822     }
823 
824     return;
825 }   /* __smoothTileTypes() */
```

4.6.3.23 assess()

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

Method to assess the resource of the selected tile.

```
1475 {
1476     HexTile* selected_tile_ptr = this->__getSelectedTile();
1477     if (selected_tile_ptr != NULL) {
1478         selected_tile_ptr->assess();
1479     }
1480
1481     return;
1482 }   /* assess() */
```

4.6.3.24 clear()

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

Method to clear the hex map.

```
1754 {
1755     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1756     std::map<double, HexTile*>::iterator hex_map_iter_y;
1757     for (
1758         hex_map_iter_x = this->hex_map.begin();
1759         hex_map_iter_x != this->hex_map.end();
1760         hex_map_iter_x++
1761     ) {
1762         for (
1763             hex_map_iter_y = hex_map_iter_x->second.begin();
1764             hex_map_iter_y != hex_map_iter_x->second.end();
1765             hex_map_iter_y++
1766         ) {
1767             delete hex_map_iter_y->second;
1768         }
1769     }
1770     this->hex_map.clear();
1771
1772     this->tile_position_x_vec.clear();
1773     this->tile_position_y_vec.clear();
1774     this->border_tiles_vec.clear();
1775
1776     return;
1777 }   /* clear() */
```

4.6.3.25 draw()

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

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

```
1673 {
1674     //  1. draw background
1675     sf::Color glass_screen_colour = this->glass_screen.getFillColor();
1676     glass_screen_colour.a = 255;
1677     this->glass_screen.setFillColor(glass_screen_colour);
1678
1679     this->render_window_ptr->draw(this->glass_screen);
1680
1681     //  2. draw tiles (other than the selected tile) in drawing order
1682     for (size_t i = 0; i < this->hex_draw_order_vec.size(); i++) {
1683         if (not this->hex_draw_order_vec[i]->is_selected) {
1684             this->hex_draw_order_vec[i]->draw();
1685         }
1686     }
1687
1688     //  3. draw total production / dispatch overlay
1689     if (this->settlement_position_logged) {
1690         this->__drawTotalDispatch();
1691     }
1692
1693     //  4. draw selected tile
1694     HexTile* selected_tile_ptr = this->__getSelectedTile();
1695     if (selected_tile_ptr != NULL) {
```

```
1696         selected_tile_ptr->draw();
1697
1698         if (
1699             (selected_tile_ptr->has_improvement) and
1700             (selected_tile_ptr->tile_improvement_ptr->tile_improvement_type ==
1701                 TileImprovementType :: SETTLEMENT)
1702         ) {
1703             this->__drawTotalDispatch();
1704         }
1705     }
1706
1707     //  5. draw resource overlay text indication
1708     if (this->show_resource) {
1709         sf::Text resource_overlay_text(
1710             "**** RENEWABLE RESOURCE OVERLAY ****",
1711             *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
1712             16
1713         );
1714
1715         resource_overlay_text.setPosition(
1716             (800 - resource_overlay_text.getLocalBounds().width) / 2,
1717             GAME_HEIGHT - 70
1718         );
1719
1720         resource_overlay_text.setFillColor(MONOCHROME_TEXT_GREEN);
1721
1722         this->render_window_ptr->draw(resource_overlay_text);
1723     }
1724
1725     //  6. draw glass screen
1726     glass_screen_colour = this->glass_screen.getFillColor();
1727     glass_screen_colour.a = 40;
1728     this->glass_screen.setFillColor(glass_screen_colour);
1729
1730     this->render_window_ptr->draw(this->glass_screen);
1731
1732     //  7. handle initial draw (tile wave animation)
1733     if (this->just_constructed) {
1734         this->__handleInitialDraw();
1735     }
1736
1737     this->frame++;
1738     return;
1739 }   /* draw() */
```

4.6.3.26 processEvent()

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

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

```
1560 {
1561     //  1. process HexTile events
1562     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1563     std::map<double, HexTile*>::iterator hex_map_iter_y;
1564     for (
1565         hex_map_iter_x = this->hex_map.begin();
1566         hex_map_iter_x != this->hex_map.end();
1567         hex_map_iter_x++
1568     ) {
1569         for (
1570             hex_map_iter_y = hex_map_iter_x->second.begin();
1571             hex_map_iter_y != hex_map_iter_x->second.end();
1572             hex_map_iter_y++
1573         ) {
1574             hex_map_iter_y->second->processEvent();
1575         }
1576     }
1577
1578     //  2. process HexMap events
1579     if (this->event_ptr->type == sf::Event::KeyPressed) {
1580         this->__handleKeyPressEvents();
1581     }
1582
1583     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1584         this->__handleMouseButtonEvents();
1585     }
1586
1587     return;
1588 }   /* processEvent() */
```

4.6.3.27 processMessage()

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

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

```
1603 {
1604     //  1. process HexTile messages
1605     std::map<double, std::map<double, HexTile*>::iterator hex_map_iter_x;
1606     std::map<double, HexTile*>::iterator hex_map_iter_y;
1607     for (
1608         hex_map_iter_x = this->hex_map.begin();
1609         hex_map_iter_x != this->hex_map.end();
1610         hex_map_iter_x++
1611     ) {
1612         for (
1613             hex_map_iter_y = hex_map_iter_x->second.begin();
1614             hex_map_iter_y != hex_map_iter_x->second.end();
1615             hex_map_iter_y++
1616         ) {
1617             hex_map_iter_y->second->processMessage();
1618         }
1619     }
1620
1621     //  2. process HexMap messages
1622     if (not this->message_hub_ptr->isEmpty(HEX_MAP_CHANNEL)) {
1623         Message hex_map_message = this->message_hub_ptr->receiveMessage(
1624             HEX_MAP_CHANNEL
1625         );
1626
1627         if (hex_map_message.subject == "assess neighbours") {
1628             HexTile* hex_ptr = this->__getSelectedTile();
1629             this->__assessNeighbours(hex_ptr);
1630
1631             std::cout « "Assess neighbours message received by " « this « std::endl;
1632             this->message_hub_ptr->popMessage(HEX_MAP_CHANNEL);
1633         }
1634     }
1635
1636     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
1637         Message game_state_message = this->message_hub_ptr->receiveMessage(
1638             GAME_STATE_CHANNEL
1639         );
1640
1641         if (game_state_message.subject == "game state") {
1642             this->demand_MWh = game_state_message.int_payload["demand_MWh"];
1643
1644             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
1645
1646             std::cout « "Game state message read and passed by " « this «
1647                 " (demand: " « this->demand_MWh « " MWh)" « std::endl;
1648         }
1649     }
1650
1651     //  3. log settlement position (if applicable)
1652     if (not this->settlement_position_logged) {
1653         this->__logSettlementPosition();
1654     }
1655
1656     return;
1657 }   /* processMessage() */
```

4.6.3.28 reroll()

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

Method to re-roll the hex map.

```
1497 {
1498     this->clear();
1499     this->__assembleHexMap();
1500
1501     return;
1502 }   /* reroll() */
```

4.6.3.29 toggleResourceOverlay()

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

Method to toggle the hex map resource overlay.

```
1517 {
1518     std::map<double, std::map<double, HexTile*»::iterator hex_map_iter_x;
1519     std::map<double, HexTile*>::iterator hex_map_iter_y;
1520     for (
1521         hex_map_iter_x = this->hex_map.begin();
1522         hex_map_iter_x != this->hex_map.end();
1523         hex_map_iter_x++
1524     ) {
1525         for (
1526             hex_map_iter_y = hex_map_iter_x->second.begin();
1527             hex_map_iter_y != hex_map_iter_x->second.end();
1528             hex_map_iter_y++
1529         ) {
1530             hex_map_iter_y->second->toggleResourceOverlay();
1531         }
1532     }
1533
1534     if (this->show_resource) {
1535         this->show_resource = false;
1536         this->assets_manager_ptr->getSound("resource overlay toggle off")->play();
1537     }
1538
1539     else {
1540         this->show_resource = true;
1541         this->assets_manager_ptr->getSound("resource overlay toggle on")->play();
1542     }
1543
1544     return;
1545 }   /* 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 dalpha

```
double HexMap::dalpha
```

The change in tile alpha (for the tile wave animation).

4.6.4.4 demand_MWh

```
int HexMap::demand_MWh
```

Current energy demand [MWh].

4.6.4.5 event_ptr

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

A pointer to the event class.

4.6.4.6 frame

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

The current frame of this object.

4.6.4.7 glass_screen

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

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

4.6.4.8 hex_draw_order_vec

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

A vector of hex tiles, in drawing order.

4.6.4.9 hex_map

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

A position-indexed, nested map of hex tiles.

4.6.4.10 initial_draw_tile_idx

```
size_t HexMap::initial_draw_tile_idx
```

The current tile idx (for the initial draw tile wave animation).

4.6.4.11 just_constructed

```
bool HexMap::just_constructed
```

A boolean which indicates if the HexMap has just been constructed.

4.6.4.12 message_hub_ptr

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

A pointer to the message hub.

4.6.4.13 n_layers

```
int HexMap::n_layers
```

The number of layers in the hex map.

4.6.4.14 n_tiles

```
int HexMap::n_tiles
```

The number of tiles in the hex map.

4.6.4.15 position_x

```
double HexMap::position_x
```

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

4.6.4.16 position_y

```
double HexMap::position_y
```

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

4.6.4.17 render_window_ptr

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

A pointer to the render window.

4.6.4.18 settlement_position_logged

```
bool HexMap::settlement_position_logged
```

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

4.6.4.19 settlement_position_x

```
double HexMap::settlement_position_x
```

The x position of the settlement.

4.6.4.20 settlement_position_y

```
double HexMap::settlement_position_y
```

The y position of the settlement.

4.6.4.21 show_resource

```
bool HexMap::show_resource
```

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

4.6.4.22 tile_position_x_vec

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

A vector of tile x positions.

4.6.4.23 tile_position_y_vec

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

A vector of tile y position.

4.6.4.24 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:

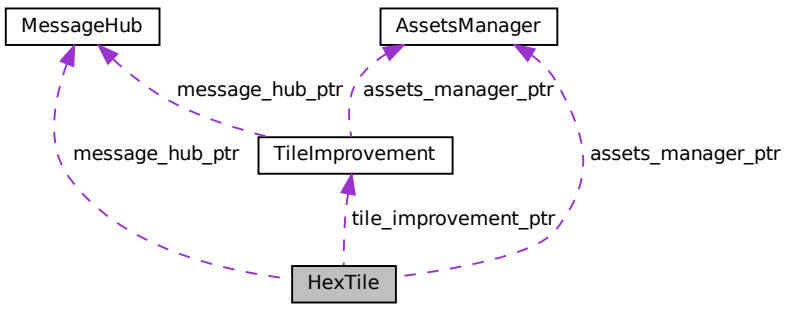
- header/HexMap.h
- source/HexMap.cpp

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. DEPRECATED.
- 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.

```
2334 {
2335     //  1. set attributes
2336
2337     //  1.1. private
2338     this->event_ptr = event_ptr;
2339     this->render_window_ptr = render_window_ptr;
2340
2341     this->assets_manager_ptr = assets_manager_ptr;
2342     this->message_hub_ptr = message_hub_ptr;
2343
2344     //  1.2. public
2345     this->show_node = false;
2346     this->show_resource = false;
2347     this->resource_assessed = false;
2348     this->resource_assessment = false;
2349     this->is_selected = false;
2350     this->draw_explosion = false;
2351
2352     this->decoration_cleared = false;
2353     this->has_improvement = false;
2354     this->tile_improvement_ptr = NULL;
2355
2356     this->build_menu_open = false;
2357
2358     this->explosion_frame = 0;
2359
2360     this->frame = 0;
2361     this->credits = 0;
2362
2363     this->scrap_improvement_frame = 0;
2364
```

```
2365     this->position_x = position_x;
2366     this->position_y = position_y;
2367
2368     this->major_radius = 32;
2369     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2370
2371     this->game_phase = "build settlement";
2372
2373     //  2. set up and position drawable attributes
2374     this->__setUpNodeSprite();
2375     this->__setUpTileSprite();
2376     this->__setUpSelectOutlineSprite();
2377     this->__setUpResourceChipSprite();
2378     this->__setResourceText();
2379     this->__setUpMagnifyingGlassSprite();
2380     this->__setUpTileExplosionReel();
2381
2382     //  3. set tile type and resource (default to none type and average)
2383     this->setTileType(TileType :: NONE_TYPE);
2384     this->setTileResource(TileResource :: AVERAGE);
2385
2386     std::cout « "HexTile constructed at " « this « std::endl;
2387
2388     return;
2389 }   /* HexTile() */
```

4.7.2.2 ~HexTile()

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

Destructor for the HexTile class.

```
2957 {
2958     if (this->tile_improvement_ptr != NULL) {
2959         delete this->tile_improvement_ptr;
2960     }
2961
2962     std::cout « "HexTile at " « this « " destroyed" « std::endl;
2963
2964     return;
2965 }   /* ~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.

```
1411 {
1412     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1413
1414     if (this->credits < build_cost) {
1415         std::cout « "Cannot build diesel generator: insufficient credits (need "
1416             « build_cost « " K)" « std::endl;
1417
1418         this->__sendInsufficientCreditsMessage();
1419         return;
1420     }
1421
1422     this->tile_improvement_ptr = new DieselGenerator(
1423         this->position_x,
1424         this->position_y,
1425         this->tile_resource,
```

```
1426         this->event_ptr,
1427         this->render_window_ptr,
1428         this->assets_manager_ptr,
1429         this->message_hub_ptr
1430     );
1431
1432     this->has_improvement = true;
1433     this->__closeBuildMenu();
1434
1435     if (not this->resource_assessed) {
1436         this->assess();
1437     }
1438
1439     this->__sendCreditsSpentMessage(build_cost);
1440     this->__sendTileStateMessage();
1441     this->__sendGameStateRequest();
1442
1443     return;
1444 }   /* __buildDieselGenerator() */
```

4.7.3.2 __buildEnergyStorage()

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

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

```
1679 {
1680     /*
1681     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1682
1683     if (this->credits < build_cost) {
1684         std::cout « "Cannot build energy storage system: insufficient credits (need "
1685             « build_cost « " K)" « std::endl;
1686
1687         this->__sendInsufficientCreditsMessage();
1688         return;
1689     }
1690
1691     this->tile_improvement_ptr = new EnergyStorageSystem(
1692         this->position_x,
1693         this->position_y,
1694         this->event_ptr,
1695         this->render_window_ptr,
1696         this->assets_manager_ptr,
1697         this->message_hub_ptr
1698     );
1699
1700     this->has_improvement = true;
1701     this->__closeBuildMenu();
1702
1703     if (not this->resource_assessed) {
1704         this->assess();
1705     }
1706
1707     this->__sendCreditsSpentMessage(build_cost);
1708     this->__sendTileStateMessage();
1709     this->__sendGameStateRequest();
1710     */
1711     return;
1712 }   /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

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

Helper method to build a settlement on this tile.

```
1364 {
1365     if (this->credits < BUILD_SETTLEMENT_COST) {
```

```
1366         std::cout « "Cannot build settlement: insufficient credits (need "
1367             « BUILD_SETTLEMENT_COST « " K)" « std::endl;
1368 
1369         this->__sendInsufficientCreditsMessage();
1370         return;
1371     }
1372 
1373     this->__clearDecoration();
1374 
1375     this->tile_improvement_ptr = new Settlement(
1376         this->position_x,
1377         this->position_y,
1378         this->tile_resource,
1379         this->event_ptr,
1380         this->render_window_ptr,
1381         this->assets_manager_ptr,
1382         this->message_hub_ptr
1383     );
1384 
1385     this->has_improvement = true;
1386 
1387     this->assess();
1388     this->__sendAssessNeighboursMessage();
1389 
1390     this->__sendUpdateGamePhaseMessage("system management");
1391     this->__sendCreditsSpentMessage(BUILD_SETTLEMENT_COST);
1392     this->__sendTileStateMessage();
1393     this->__sendGameStateRequest();
1394 
1395     return;
1396 }   /* __buildSettlement() */
```

4.7.3.4 __buildSolarPV()

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

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

```
1459 {
1460     int build_cost = SOLAR_PV_BUILD_COST;
1461 
1462     if (this->tile_type == TileType :: LAKE) {
1463         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1464     }
1465 
1466     if (this->credits < build_cost) {
1467         std::cout « "Cannot build solar PV array: insufficient credits (need "
1468             « build_cost « " K)" « std::endl;
1469 
1470         this->__sendInsufficientCreditsMessage();
1471         return;
1472     }
1473 
1474     this->tile_improvement_ptr = new SolarPV(
1475         this->position_x,
1476         this->position_y,
1477         this->tile_resource,
1478         this->event_ptr,
1479         this->render_window_ptr,
1480         this->assets_manager_ptr,
1481         this->message_hub_ptr
1482     );
1483 
1484     this->has_improvement = true;
1485     this->__closeBuildMenu();
1486 
1487     if (not this->resource_assessed) {
1488         this->assess();
1489     }
1490 
1491     if (this->tile_type == TileType :: LAKE) {
1492         this->decoration_cleared = true;
1493         this->assets_manager_ptr->getSound("splash")->play();
1494     }
1495 
1496     this->__sendCreditsSpentMessage(build_cost);
1497     this->__sendTileStateMessage();
1498     this->__sendGameStateRequest();
```

```
1499
1500     return;
1501 }   /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

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

Helper method to build a tidal turbine on this tile.

```
1579 {
1580     int build_cost = TIDAL_TURBINE_BUILD_COST;
1581
1582     if (this->credits < build_cost) {
1583         std::cout « "Cannot build tidal turbine: insufficient credits (need "
1584             « build_cost « " K)" « std::endl;
1585
1586         this->__sendInsufficientCreditsMessage();
1587         return;
1588     }
1589
1590     this->tile_improvement_ptr = new TidalTurbine(
1591         this->position_x,
1592         this->position_y,
1593         this->tile_resource,
1594         this->event_ptr,
1595         this->render_window_ptr,
1596         this->assets_manager_ptr,
1597         this->message_hub_ptr
1598     );
1599
1600     this->has_improvement = true;
1601     this->decoration_cleared = true;
1602     this->assets_manager_ptr->getSound("splash")->play();
1603     this->__closeBuildMenu();
1604
1605     if (not this->resource_assessed) {
1606         this->assess();
1607     }
1608
1609     this->__sendCreditsSpentMessage(build_cost);
1610     this->__sendTileStateMessage();
1611     this->__sendGameStateRequest();
1612
1613     return;
1614 }   /* __buildTidalTurbine() */
```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1629 {
1630     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1631
1632     if (this->credits < build_cost) {
1633         std::cout « "Cannot build wave energy converter: insufficient credits (need "
1634             « build_cost « " K)" « std::endl;
1635
1636         this->__sendInsufficientCreditsMessage();
1637         return;
1638     }
1639
1640     this->tile_improvement_ptr = new WaveEnergyConverter(
1641         this->position_x,
1642         this->position_y,
1643         this->tile_resource,
1644         this->event_ptr,
```

```
1645         this->render_window_ptr,
1646         this->assets_manager_ptr,
1647         this->message_hub_ptr
1648     );
1649 
1650     this->has_improvement = true;
1651     this->decoration_cleared = true;
1652     this->assets_manager_ptr->getSound("splash")->play();
1653     this->__closeBuildMenu();
1654 
1655     if (not this->resource_assessed) {
1656         this->assess();
1657     }
1658 
1659     this->__sendCreditsSpentMessage(build_cost);
1660     this->__sendTileStateMessage();
1661     this->__sendGameStateRequest();
1662 
1663     return;
1664 }   /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1516 {
1517     int build_cost = WIND_TURBINE_BUILD_COST;
1518 
1519     if (
1520         (this->tile_type == TileType :: LAKE) or
1521         (this->tile_type == TileType :: OCEAN)
1522     ) {
1523         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1524     }
1525 
1526     if (this->credits < build_cost) {
1527         std::cout « "Cannot build wind turbine: insufficient credits (need "
1528             « build_cost « " K)" « std::endl;
1529 
1530         this->__sendInsufficientCreditsMessage();
1531         return;
1532     }
1533 
1534     this->tile_improvement_ptr = new WindTurbine(
1535         this->position_x,
1536         this->position_y,
1537         this->tile_resource,
1538         this->event_ptr,
1539         this->render_window_ptr,
1540         this->assets_manager_ptr,
1541         this->message_hub_ptr
1542     );
1543 
1544     this->has_improvement = true;
1545     this->__closeBuildMenu();
1546 
1547     if (not this->resource_assessed) {
1548         this->assess();
1549     }
1550 
1551     if (
1552         (this->tile_type == TileType :: LAKE) or
1553         (this->tile_type == TileType :: OCEAN)
1554     ) {
1555         this->decoration_cleared = true;
1556         this->assets_manager_ptr->getSound("splash")->play();
1557     }
1558 
1559     this->__sendCreditsSpentMessage(build_cost);
1560     this->__sendTileStateMessage();
1561     this->__sendGameStateRequest();
1562 
1563     return;
1564 }   /* __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.

```
1339 {
1340     if (not this->build_menu_open) {
1341         return;
1342     }
1343 
1344     this->build_menu_open = false;
1345     this->assets_manager_ptr->getSound("build menu close")->play();
1346 
1347     return;
1348 }   /* __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.

```
1829 {
1830     std::string coords_substring = "TILE COORDS:  (";
1831     coords_substring += std::to_string(int(this->position_x - 400));
1832     coords_substring += ", ";
1833     coords_substring += std::to_string(int(this->position_y - 400));
1834     coords_substring += ")\n";
1835 
1836     return coords_substring;
1837 }   /* __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.

```
1988 {
1989     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1990 
1991     if (this->has_improvement) {
1992         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1993         improvement_substring += "\n";
1994     }
1995 
1996     else {
1997         improvement_substring += "NONE\n";
1998     }
1999 
2000     return improvement_substring;
2001 }   /* __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.

```
2018 {
2019     //                  32 char x 17 line console "--------------------------------\n";
2020     std::string options_substring               = "     **** TILE OPTIONS ****     \n";
2021     options_substring                          += "                                \n";
2022 
2023     if (this->game_phase == "build settlement") {
2024         if (
2025             (this->tile_type != TileType :: OCEAN) and
2026             (this->tile_type != TileType :: LAKE)
2027         ) {
2028             options_substring += "[B]:  BUILD SETTLEMENT (";
2029             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
2030             options_substring += " K)\n";
2031         }
```

```
2032     }
2033
2034
2035     else if (this->game_phase == "system management") {
2036         if (this->has_improvement) {
2037             options_substring.clear();
2038             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2039         }
2040
2041
2042         else if (not this->resource_assessed) {
2043             options_substring += "[A]:  ASSESS RESOURCE (";
2044             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2045             options_substring += " K)\n";
2046         }
2047
2048
2049         else if (
2050             (not this->decoration_cleared) and
2051             (this->tile_type != TileType :: OCEAN) and
2052             (this->tile_type != TileType :: LAKE)
2053         ) {
2054             options_substring += "[C]:  CLEAR TILE (";
2055
2056             switch (this->tile_type) {
2057                 case (TileType :: FOREST): {
2058                     options_substring += std::to_string(CLEAR_FOREST_COST);
2059
2060                     break;
2061                 }
2062
2063
2064                 case (TileType :: MOUNTAINS): {
2065                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2066
2067                     break;
2068                 }
2069
2070
2071                 case (TileType :: PLAINS): {
2072                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2073
2074                     break;
2075                 }
2076
2077
2078                 default: {
2079                     //do nothing!
2080
2081                     break;
2082                 }
2083             }
2084
2085             options_substring += " K)\n";
2086         }
2087
2088
2089         else if (
2090             (this->decoration_cleared) or
2091             (this->tile_type == TileType :: OCEAN) or
2092             (this->tile_type == TileType :: LAKE)
2093         ) {
2094             options_substring += "[B]: OPEN BUILD MENU\n";
2095         }
2096     }
2097
2098
2099     else if (this->game_phase == "victory") {
2100         options_substring                       += "       **** VICTORY ****        \n";
2101     }
2102
2103
2104     else {
2105         options_substring                       += "        **** LOSS ****          \n";
2106     }
2107
2108     return options_substring;
2109 }   /* __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.

```
1918 {
1919     std::string resource_substring = "TILE RESOURCE:     ";
1920 
1921     if (this->resource_assessed) {
1922         switch (this->tile_resource) {
1923             case (TileResource :: POOR): {
1924                 resource_substring += "POOR\n";
1925 
1926                 break;
1927             }
1928 
1929 
1930             case (TileResource ::BELOW_AVERAGE): {
1931                 resource_substring += "BELOW AVERAGE\n";
1932 
1933                 break;
1934             }
1935 
1936 
1937             case (TileResource :: AVERAGE): {
1938                 resource_substring += "AVERAGE\n";
1939 
1940                 break;
1941             }
1942 
1943 
1944             case (TileResource :: ABOVE_AVERAGE): {
1945                 resource_substring += "ABOVE AVERAGE\n";
1946 
1947                 break;
1948             }
1949 
1950 
1951             case (TileResource :: GOOD): {
1952                 resource_substring += "GOOD\n";
1953 
1954                 break;
1955             }
1956 
1957 
1958             default: {
1959                 resource_substring += "???\n";
1960 
1961                 break;
1962             }
1963         }
1964     }
1965 
1966     else {
1967         resource_substring += "???\n";
1968     }
1969 
1970     return resource_substring;
1971 }   /* __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.

```
1854 {
1855     std::string type_substring = "TILE TYPE:         ";
1856 
1857     switch (this->tile_type) {
1858         case (TileType :: FOREST): {
1859             type_substring += "FOREST\n";
1860 
1861             break;
1862         }
1863 
1864 
1865         case (TileType :: LAKE): {
1866             type_substring += "LAKE\n";
1867 
1868             break;
1869         }
1870 
1871 
1872         case (TileType :: MOUNTAINS): {
1873             type_substring += "MOUNTAINS\n";
1874 
1875             break;
1876         }
1877 
1878 
1879         case (TileType :: OCEAN): {
1880             type_substring += "OCEAN\n";
1881 
1882             break;
1883         }
1884 
1885 
1886         case (TileType :: PLAINS): {
1887             type_substring += "PLAINS\n";
1888 
1889             break;
1890         }
1891 
1892 
1893         default: {
1894             type_substring += "???\n";
1895 
1896             break;
1897         }
1898     }
1899 
1900     return type_substring;
1901 }   /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
876 {
877     if (not this->is_selected) {
878         return;
879     }
880 
881 
882     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
883         this->__setIsSelected(false);
884     }
885 
886 
887     if (this->build_menu_open) {
888         switch (this->tile_type) {
889             case (TileType :: FOREST): {
890                 switch (this->event_ptr->key.code) {
891                     case (sf::Keyboard::D): {
892                         this->__buildDieselGenerator();
893 
894                         break;
```

```
895                     }
896
897
898                     case (sf::Keyboard::S): {
899                         this->__buildSolarPV();
900
901                         break;
902                     }
903
904
905                     case (sf::Keyboard::W): {
906                         this->__buildWindTurbine();
907
908                         break;
909                     }
910
911
912                     case (sf::Keyboard::E): {
913                         this->__buildEnergyStorage();
914
915                         break;
916                     }
917
918
919                     default: {
920                         // do nothing!
921
922                         break;
923                     }
924                 }
925
926                 break;
927             }
928
929
930             case (TileType :: LAKE): {
931                 switch (this->event_ptr->key.code) {
932                     case (sf::Keyboard::S): {
933                         this->__buildSolarPV();
934
935                         break;
936                     }
937
938
939                     case (sf::Keyboard::W): {
940                         this->__buildWindTurbine();
941
942                         break;
943                     }
944
945
946                     default: {
947                         // do nothing!
948
949                         break;
950                     }
951                 }
952
953                 break;
954             }
955
956
957             case (TileType :: MOUNTAINS): {
958                 switch (this->event_ptr->key.code) {
959                     case (sf::Keyboard::D): {
960                         this->__buildDieselGenerator();
961
962                         break;
963                     }
964
965
966                     case (sf::Keyboard::S): {
967                         this->__buildSolarPV();
968
969                         break;
970                     }
971
972
973                     case (sf::Keyboard::W): {
974                         this->__buildWindTurbine();
975
976                         break;
977                     }
978
979
980                     case (sf::Keyboard::E): {
981                         this->__buildEnergyStorage();
```

```
982
983                         break;
984                     }
985
986
987                     default: {
988                         // do nothing!
989
990                         break;
991                     }
992                 }
993
994                 break;
995             }
996
997
998             case (TileType :: OCEAN): {
999                 switch (this->event_ptr->key.code) {
1000                     case (sf::Keyboard::W): {
1001                         this->__buildWindTurbine();
1002
1003                         break;
1004                     }
1005
1006
1007                     case (sf::Keyboard::T): {
1008                         this->__buildTidalTurbine();
1009
1010                         break;
1011                     }
1012
1013
1014                     case (sf::Keyboard::A): {
1015                         this->__buildWaveEnergyConverter();
1016
1017                         break;
1018                     }
1019
1020
1021                     default: {
1022                         // do nothing!
1023
1024                         break;
1025                     }
1026                 }
1027
1028                 break;
1029             }
1030
1031
1032             case (TileType :: PLAINS): {
1033                 switch (this->event_ptr->key.code) {
1034                     case (sf::Keyboard::D): {
1035                         this->__buildDieselGenerator();
1036
1037                         break;
1038                     }
1039
1040
1041                     case (sf::Keyboard::S): {
1042                         this->__buildSolarPV();
1043
1044                         break;
1045                     }
1046
1047
1048                     case (sf::Keyboard::W): {
1049                         this->__buildWindTurbine();
1050
1051                         break;
1052                     }
1053
1054
1055                     case (sf::Keyboard::E): {
1056                         this->__buildEnergyStorage();
1057
1058                         break;
1059                     }
1060
1061
1062                     default: {
1063                         // do nothing!
1064
1065                         break;
1066                     }
1067                 }
1068
```

```
1069                 break;
1070             }
1071
1072
1073             default: {
1074                 //do nothing!
1075
1076                 break;
1077             }
1078         }
1079     }
1080
1081
1082     if (this->game_phase == "build settlement") {
1083         if (
1084             (this->tile_type != TileType :: OCEAN) and
1085             (this->tile_type != TileType :: LAKE)
1086         ) {
1087             if (this->event_ptr->key.code == sf::Keyboard::B) {
1088                 this->__buildSettlement();
1089             }
1090         }
1091     }
1092
1093
1094     else if (this->game_phase == "system management") {
1095         if (this->has_improvement) {
1096             if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
     {
1097                 if (this->event_ptr->key.code == sf::Keyboard::P) {
1098                     this->__scrapImprovement();
1099                 }
1100             }
1101
1102             /*
1103              * All other inputs will be caught and handled by
1104              *   this->tile_improvement_ptr->processEvent()
1105              */
1106         }
1107
1108
1109         else if (not this->resource_assessed) {
1110             if (this->event_ptr->key.code == sf::Keyboard::A) {
1111                 if (this->credits < RESOURCE_ASSESSMENT_COST) {
1112                     std::cout « "Cannot assess resource: insufficient credits (need "
1113                         « RESOURCE_ASSESSMENT_COST « " K)" « std::endl;
1114
1115                     this->__sendInsufficientCreditsMessage();
1116                 }
1117
1118                 else {
1119                     this->assess();
1120                     this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1121                     this->__sendTileStateMessage();
1122                     this->__sendGameStateRequest();
1123                 }
1124             }
1125         }
1126
1127
1128         else if (
1129             (not this->decoration_cleared) and
1130             (this->tile_type != TileType :: OCEAN) and
1131             (this->tile_type != TileType :: LAKE)
1132         ) {
1133             if (this->event_ptr->key.code == sf::Keyboard::C) {
1134                 int clear_cost = 0;
1135
1136                 switch (this->tile_type) {
1137                     case (TileType :: FOREST): {
1138                         clear_cost = CLEAR_FOREST_COST;
1139
1140                         break;
1141                     }
1142
1143
1144                     case (TileType :: MOUNTAINS): {
1145                         clear_cost = CLEAR_MOUNTAINS_COST;
1146
1147                         break;
1148                     }
1149
1150
1151                     case (TileType :: PLAINS): {
1152                         clear_cost = CLEAR_PLAINS_COST;
1153
1154                         break;
```

```
1155                     }
1156
1157
1158                     default: {
1159                         // do nothing!
1160
1161                         break;
1162                     }
1163                 }
1164
1165                 if (this->credits < clear_cost) {
1166                     std::cout « "Cannot clear tile: insufficient credits (need "
1167                         « clear_cost « " K)" « std::endl;
1168
1169                     this->__sendInsufficientCreditsMessage();
1170                 }
1171
1172                 else {
1173                     this->__clearDecoration();
1174                     this->__sendCreditsSpentMessage(clear_cost);
1175                     this->__sendTileStateMessage();
1176                     this->__sendGameStateRequest();
1177                 }
1178             }
1179         }
1180
1181
1182         else if (
1183             (this->decoration_cleared) or
1184             (this->tile_type == TileType :: OCEAN) or
1185             (this->tile_type == TileType :: LAKE)
1186         ) {
1187             if (this->event_ptr->key.code == sf::Keyboard::B) {
1188                 this->__openBuildMenu();
1189             }
1190         }
1191     }
1192
1193     return;
1194 }   /* __handleKeyPressEvents() */
```

4.7.3.16 __handleKeyReleaseEvents()

```
void HexTile::__handleKeyReleaseEvents (
             void  )  [private]
1200 {
1201     if (not this->is_selected) {
1202         return;
1203     }
1204
1205
1206     switch (this->event_ptr->key.code) {
1207         case (sf::Keyboard::P): {
1208             if (this->has_improvement) {
1209                 this->scrap_improvement_frame = 0;
1210
1211                 if (
1212                     this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1213                 ) {
1214                     this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1215                         sf::Color(255, 255, 255, 255)
1216                     );
1217                 }
1218
1219                 else {
1220                     for (
1221                         size_t i = 0;
1222                         i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1223                         i++
1224                     ) {
1225                         this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1226                             sf::Color(255, 255, 255, 255)
1227                         );
1228                     }
1229                 }
1230             }
1231
1232
1233             break;
```

```
1234         }
1235
1236
1237         default: {
1238             // do nothing!
1239
1240             break;
1241         }
1242     }
1243
1244     /*
1245     if (this->event_ptr->key.code == sf::Keyboard::P) {
1246
1247     }
1248     */
1249
1250     return;
1251 }   /* __handleKeyReleaseEvents() */
```

4.7.3.17 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
1264 {
1265     switch (this->event_ptr->mouseButton.button) {
1266         case (sf::Mouse::Left): {
1267             if (this->__isClicked()) {
1268                 std::cout « "Tile (" « this->position_x « ", " «
1269                     this->position_y « ") was selected" « std::endl;
1270
1271                 this->__setIsSelected(true);
1272
1273                 this->__sendTileSelectedMessage();
1274                 this->__sendTileStateMessage();
1275             }
1276
1277             else {
1278                 this->__setIsSelected(false);
1279             }
1280
1281             break;
1282         }
1283
1284
1285         case (sf::Mouse::Right): {
1286             this->__setIsSelected(false);
1287
1288             break;
1289         }
1290
1291
1292         default: {
1293             // do nothing!
1294
1295             break;
1296         }
1297     }
1298
1299     return;
1300 }   /* __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::Vector2f mouse_position = this->render_window_ptr->mapPixelToCoords(
844         sf::Mouse::getPosition(*(this->render_window_ptr))
845     );
846 
847     double mouse_x = mouse_position.x;
848     double mouse_y = mouse_position.y;
849 
850     double distance = sqrt(
851         pow(this->position_x - mouse_x, 2) +
852         pow(this->position_y - mouse_y, 2)
853     );
854 
855     if (distance < this->minor_radius) {
856         return true;
857     }
858     else {
859         return false;
860     }
861 }   /* __isClicked() */
```

4.7.3.19 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

```
1315 {
1316     if (this->build_menu_open) {
1317         return;
1318     }
1319 
1320     this->build_menu_open = true;
1321     this->assets_manager_ptr->getSound("build menu open")->play();
1322 
1323     return;
1324 }   /* __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.

```
1728 {
1729     //  1. implement key hold confirmation
1730     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1731         double colour_scalar =
1732             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1733 
1734         if (
1735             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1736         ) {
1737             this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1738                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1739             );
1740         }
1741 
1742         else {
1743             for (
1744                 size_t i = 0;
1745                 i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1746                 i++
1747             ) {
```

```
1748                 this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1749                     sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1750                 );
1751             }
1752         }
1753 
1754         this->scrap_improvement_frame += 4;
1755     }
1756 
1757 
1758     //  2. carry out scrapping
1759     else {
1760         this->draw_explosion = true;
1761         this->assets_manager_ptr->getSound("clear non-mountains tile")->play();
1762 
1763         if (this->tile_improvement_ptr->production_menu_open) {
1764             this->tile_improvement_ptr->production_menu_open = false;
1765             this->assets_manager_ptr->getSound("build menu close")->play();
1766         }
1767 
1768         delete this->tile_improvement_ptr;
1769         this->tile_improvement_ptr = NULL;
1770 
1771         this->has_improvement = false;
1772 
1773         this->scrap_improvement_frame = 0;
1774 
1775         if (
1776             (this->tile_type == TileType :: LAKE) or
1777             (this->tile_type == TileType :: OCEAN)
1778         ) {
1779             this->decoration_cleared = false;
1780         }
1781 
1782         this->__sendCreditsSpentMessage(SCRAP_COST);
1783         this->__sendTileStateMessage();
1784         this->__sendGameStateRequest();
1785     }
1786 
1787     return;
1788 }   /* __scrapImprovement() */
```

4.7.3.21 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2165 {
2166     Message assess_neighbours_message;
2167 
2168     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2169     assess_neighbours_message.subject = "assess neighbours";
2170 
2171     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2172 
2173     std::cout « "Assess neighbours message sent by " « this « std::endl;
2174 
2175     return;
2176 }   /* __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.

```
2248 {
2249     Message credits_spent_message;
2250 
2251     credits_spent_message.channel = GAME_CHANNEL;
2252     credits_spent_message.subject = "credits spent";
2253 
2254     credits_spent_message.int_payload["credits spent"] = credits_spent;
2255 
2256     this->message_hub_ptr->sendMessage(credits_spent_message);
2257 
2258     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2259         << std::endl;
2260     return;
2261 }   /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2191 {
2192     Message game_state_request;
2193 
2194     game_state_request.channel = GAME_CHANNEL;
2195     game_state_request.subject = "state request";
2196 
2197     this->message_hub_ptr->sendMessage(game_state_request);
2198 
2199     std::cout << "Game state request message sent by " << this << std::endl;
2200     return;
2201 }   /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2276 {
2277     Message insufficient_credits_message;
2278 
2279     insufficient_credits_message.channel = GAME_CHANNEL;
2280     insufficient_credits_message.subject = "insufficient credits";
2281 
2282     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2283 
2284     std::cout << "Insufficient credits message sent by " << this << std::endl;
2285 
2286     return;
2287 }   /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1803 {
1804     Message tile_selected_message;
1805 
1806     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1807     tile_selected_message.subject = "tile selected";
1808 
1809     this->message_hub_ptr->sendMessage(tile_selected_message);
1810 
1811     return;
1812 }   /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2124 {
2125     Message tile_state_message;
2126 
2127     tile_state_message.channel = TILE_STATE_CHANNEL;
2128     tile_state_message.subject = "tile state";
2129 
2130 
2131     //                  32 char x 17 line console "--------------------------------\n";
2132     std::string console_string                    = "      **** TILE INFO ****       \n";
2133 
2134     console_string                               += this->__getTileCoordsSubstring();
2135     console_string                               += "                                \n";
2136 
2137     console_string                               += this->__getTileTypeSubstring();
2138     console_string                               += this->__getTileResourceSubstring();
2139     console_string                               += this->__getTileImprovementSubstring();
2140     console_string                               += "                                \n";
2141 
2142     console_string                               += this->__getTileOptionsSubstring();
2143 
2144     tile_state_message.string_payload["console string"] = console_string;
2145 
2146     this->message_hub_ptr->sendMessage(tile_state_message);
2147 
2148     std::cout « "Tile state message sent by " « this « std::endl;
2149     return;
2150 }   /* __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.

```
2218 {
2219     Message update_game_phase_message;
2220
2221     update_game_phase_message.channel = GAME_CHANNEL;
2222     update_game_phase_message.subject = "update game phase";
2223
2224     update_game_phase_message.string_payload["game phase"] = game_phase;
2225
2226     this->message_hub_ptr->sendMessage(update_game_phase_message);
2227
2228     std::cout « "Update game phase message sent by " « this « std::endl;
2229
2230     return;
2231 }   /* __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: 200 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.

```
2710 {
2711     this->resource_assessed = true;
2712     this->resource_assessment = true;
2713
2714     this->assets_manager_ptr->getSound("resource assessment")->play();
2715
2716     this->__setResourceText();
2717     this->__sendTileStateMessage();
2718
2719     return;
2720 }   /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2588 {
2589     switch (this->tile_type) {
2590         case (TileType :: FOREST): {
2591             this->tile_decoration_sprite.setTexture(
2592                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2593             );
2594
2595             break;
2596         }
2597
2598         case (TileType :: LAKE): {
2599             this->tile_decoration_sprite.setTexture(
2600                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2601             );
2602
2603             break;
2604         }
2605
2606         case (TileType :: MOUNTAINS): {
2607             this->tile_decoration_sprite.setTexture(
2608                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2609             );
2610
2611             break;
2612         }
2613
2614         case (TileType :: OCEAN): {
2615             this->tile_decoration_sprite.setTexture(
2616                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2617             );
2618
2619             break;
2620         }
2621
2622         case (TileType :: PLAINS): {
2623             this->tile_decoration_sprite.setTexture(
2624                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2625             );
2626
2627             break;
2628         }
2629
2630         default: {
2631             // do nothing!
2632
2633             break;
2634         }
2635     }
2636
2637
2638     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```
2639         this->tile_decoration_sprite.setOrigin(
2640             this->tile_decoration_sprite.getLocalBounds().width / 2,
2641             this->tile_decoration_sprite.getLocalBounds().height / 2
2642         );
2643 
2644         this->tile_decoration_sprite.setPosition(
2645             this->position_x,
2646             this->position_y
2647         );
2648 
2649         if ((double)rand() / RAND_MAX > 0.5) {
2650             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2651         }
2652     }
2653 
2654     else {
2655         this->tile_decoration_sprite.setOrigin(
2656             this->tile_decoration_sprite.getLocalBounds().width / 2,
2657             this->tile_decoration_sprite.getLocalBounds().height
2658         );
2659 
2660         this->tile_decoration_sprite.setPosition(
2661             this->position_x,
2662             this->position_y + 12
2663         );
2664 
2665         if ((double)rand() / RAND_MAX > 0.5) {
2666             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2667         }
2668     }
2669 
2670     return;
2671 }   /* 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.

```
2851 {
2852     //  1. draw hex
2853     this->render_window_ptr->draw(this->tile_sprite);
2854 
2855     //  2. draw node
2856     if (this->show_node) {
2857         this->render_window_ptr->draw(this->node_sprite);
2858     }
2859 
2860     //  3. draw tile decoration
2861     if (not this->decoration_cleared) {
2862         this->render_window_ptr->draw(this->tile_decoration_sprite);
2863     }
2864 
2865     //  4. draw selection outline
2866     if (this->is_selected) {
2867         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2868 
2869         outline_colour.a =
2870             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2871 
2872         this->select_outline_sprite.setOutlineColor(outline_colour);
2873 
2874         this->render_window_ptr->draw(this->select_outline_sprite);
2875     }
2876 
2877     //  5. draw tile improvement
2878     if (this->has_improvement) {
2879         if (not this->tile_improvement_ptr->just_built) {
2880             this->tile_improvement_ptr->draw();
2881         }
2882     }
2883 
2884     //  6. draw resource
2885     if (this->show_resource) {
2886         this->render_window_ptr->draw(this->resource_chip_sprite);
2887         this->render_window_ptr->draw(this->resource_text);
2888     }
```

```
2889
2890     //  7. draw resource assessment notification
2891     if (this->resource_assessment) {
2892         int alpha = this->magnifying_glass_sprite.getColor().a;
2893
2894         alpha -= 0.05 * FRAMES_PER_SECOND;
2895         if (alpha < 0) {
2896             alpha = 0;
2897             this->resource_assessment = false;
2898         }
2899
2900         this->magnifying_glass_sprite.setColor(
2901             sf::Color(255, 255, 255, alpha)
2902         );
2903
2904         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2905     }
2906
2907     //  8. draw explosion, then settlement placement
2908     if (this->draw_explosion) {
2909         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2910
2911         if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2912             this->explosion_frame++;
2913         }
2914
2915         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2916             this->draw_explosion = false;
2917             this->explosion_frame = 0;
2918         }
2919     }
2920
2921     else if (this->has_improvement) {
2922         if (this->tile_improvement_ptr->just_built) {
2923             this->tile_improvement_ptr->draw();
2924         }
2925     }
2926
2927     //  9. build menu
2928     if (this->build_menu_open) {
2929         this->render_window_ptr->draw(this->build_menu_backing);
2930         this->render_window_ptr->draw(this->build_menu_backing_text);
2931
2932         for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2933             for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2934                 this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2935             }
2936             this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2937         }
2938     }
2939
2940     this->frame++;
2941     return;
2942 }   /* draw() */
```

4.7.3.47 processEvent()

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

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

```
2735 {
2736     //  1. process TileImprovement events
2737     if (
2738         this->is_selected and
2739         this->tile_improvement_ptr != NULL
2740     ) {
2741         this->tile_improvement_ptr->processEvent();
2742     }
2743
2744     //  2. process HexTile events
2745     if (this->event_ptr->type == sf::Event::KeyPressed) {
2746         this->__handleKeyPressEvents();
2747     }
2748
2749     if (this->event_ptr->type == sf::Event::KeyReleased) {
2750         this->__handleKeyReleaseEvents();
2751     }
```

```
2752
2753     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2754         this->__handleMouseButtonEvents();
2755     }
2756
2757     return;
2758 }    /* processEvent() */
```

4.7.3.48 processMessage()

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

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

```
2773 {
2774     //  1. process TileImprovement messages
2775     if (this->tile_improvement_ptr != NULL) {
2776         this->tile_improvement_ptr->processMessage();
2777     }
2778
2779     //  2. process HexTile messages
2780     if (this->is_selected) {
2781         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2782             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2783                 TILE_STATE_CHANNEL
2784             );
2785
2786             if (tile_state_message.subject == "state request") {
2787                 this->__sendTileStateMessage();
2788
2789                 std::cout « "Tile state request received by " « this « std::endl;
2790                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2791             }
2792         }
2793
2794         std::cout « "Current credits (HexTile): " « this->credits « " K" «
2795             std::endl;
2796     }
2797
2798     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2799         Message game_state_message = this->message_hub_ptr->receiveMessage(
2800             GAME_STATE_CHANNEL
2801         );
2802
2803         if (game_state_message.subject == "game state") {
2804             this->credits = game_state_message.int_payload["credits"];
2805             this->game_phase = game_state_message.string_payload["game phase"];
2806
2807             if (this->tile_improvement_ptr != NULL) {
2808                 this->tile_improvement_ptr->credits = this->credits;
2809                 this->tile_improvement_ptr->game_phase = this->game_phase;
2810
2811                 this->tile_improvement_ptr->month =
2812                     game_state_message.int_payload["month"];
2813
2814                 this->tile_improvement_ptr->demand_MWh =
2815                     game_state_message.int_payload["demand_MWh"];
2816
2817                 this->tile_improvement_ptr->demand_vec_MWh =
2818                     game_state_message.vector_payload["demand_vec_MWh"];
2819
2820                 this->tile_improvement_ptr->update();
2821             }
2822
2823             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
2824
2825             std::cout « "Game state message read and passed by " « this «
2826                 " (credits: " « this->credits « " K)" « std::endl;
2827
2828             if (this->is_selected) {
2829                 this->__sendTileStateMessage();
2830             }
2831         }
2832     }
2833
2834     return;
2835 }    /* 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].

```
2537 {
2538     //  1. check input
2539     if (input_value < 0 or input_value > 1) {
2540         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
2541         error_str += "not in the closed interval [0, 1]";
2542 
2543         #ifdef _WIN32
2544             std::cout « error_str « std::endl;
2545         #endif  /* _WIN32 */
2546 
2547         throw std::runtime_error(error_str);
2548     }
2549 
2550     //  2. convert input value to tile resource
2551     TileResource tile_resource;
2552 
2553     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2554         tile_resource = TileResource :: POOR;
2555     }
2556     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2557         tile_resource = TileResource :: BELOW_AVERAGE;
2558     }
2559     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2560         tile_resource = TileResource :: AVERAGE;
2561     }
2562     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2563         tile_resource = TileResource :: ABOVE_AVERAGE;
2564     }
2565     else {
2566         tile_resource = TileResource :: GOOD;
2567     }
2568 
2569     //  3. call alternate method
2570     this->setTileResource(tile_resource);
2571 
2572     return;
2573 }   /* 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.

```
2515 {
2516     this->tile_resource = tile_resource;
2517     this->__setResourceText();
2518 
2519     return;
2520 }   /* 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].

```
2465 {
2466     //  1. check input
2467     if (input_value < 0 or input_value > 1) {
2468         std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
2469         error_str += "not in the closed interval [0, 1]";
2470 
2471         #ifdef _WIN32
2472             std::cout « error_str « std::endl;
2473         #endif  /* _WIN32 */
2474 
2475         throw std::runtime_error(error_str);
2476     }
2477 
2478     //  2. convert input value to tile type
2479     TileType tile_type;
2480 
2481     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2482         tile_type = TileType :: LAKE;
2483     }
2484     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2485         tile_type = TileType :: PLAINS;
2486     }
2487     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2488         tile_type = TileType :: FOREST;
2489     }
2490     else {
2491         tile_type = TileType :: MOUNTAINS;
2492     }
2493 
2494     //  3. call alternate method
2495     this->setTileType(tile_type);
2496 
2497     return;
2498 }   /* 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.

```
2404 {
2405     this->tile_type = tile_type;
2406 
2407     switch (this->tile_type) {
2408         case (TileType :: FOREST): {
2409             this->tile_sprite.setFillColor(FOREST_GREEN);
2410 
2411             break;
2412         }
2413 
2414         case (TileType :: LAKE): {
```

```
2415             this->tile_sprite.setFillColor(LAKE_BLUE);
2416
2417             break;
2418         }
2419
2420         case (TileType :: MOUNTAINS): {
2421             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2422
2423             break;
2424         }
2425
2426         case (TileType :: OCEAN): {
2427             this->tile_sprite.setFillColor(OCEAN_BLUE);
2428
2429             break;
2430         }
2431
2432         case (TileType :: PLAINS): {
2433             this->tile_sprite.setFillColor(PLAINS_YELLOW);
2434
2435             break;
2436         }
2437
2438         default: {
2439             // do nothing!
2440
2441             break;
2442         }
2443     }
2444
2445     this->__setUpBuildMenu();
2446
2447     return;
2448 }   /* setTileType(TileType) */
```

4.7.3.53 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
2686 {
2687     if (this->show_resource) {
2688         this->show_resource = false;
2689     }
2690     else {
2691         this->show_resource = true;
2692     }
2693
2694     return;
2695 }   /* 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:

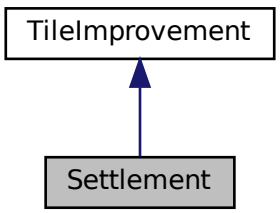
- header/ESC_core/MessageHub.h
- source/ESC_core/MessageHub.cpp

4.10 Settlement Class Reference

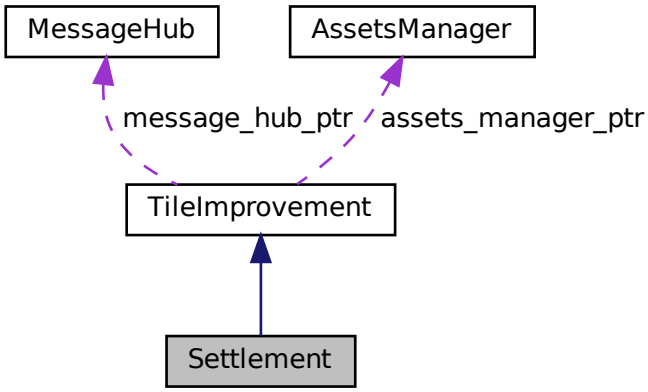
A settlement class (child class of TileImprovement).

```
#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:

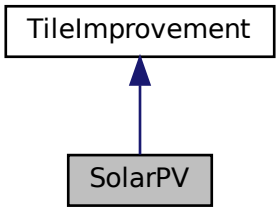
- header/Settlement.h
- source/Settlement.cpp

4.11 SolarPV Class Reference

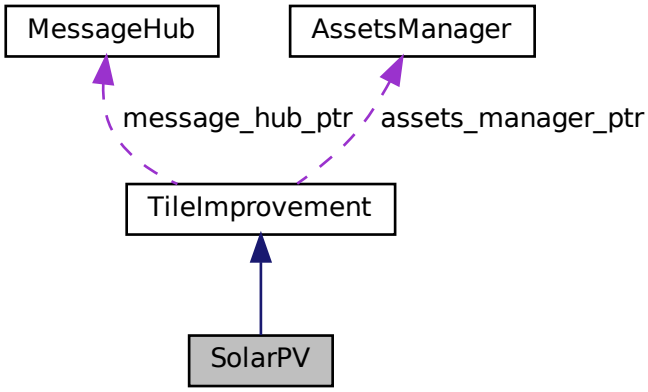
A settlement class (child class of TileImprovement).

```
#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 (by definition in the closed interval [0, 1]).
- 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.

```
777   :
778 TileImprovement(
779     position_x,
780     position_y,
781     tile_resource,
782     event_ptr,
783     render_window_ptr,
784     assets_manager_ptr,
785     message_hub_ptr
786 )
787 {
788     //  1. set attributes
789
790     //  1.1. private
791     //...
792
793     //  1.2. public
794     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
795
796     this->is_running = false;
797
798     this->health = 100;
799
800     this->capacity_kW = 100;
801     this->upgrade_level = 1;
802
803     this->storage_kWh = 0;
804     this->storage_level = 0;
805
806     this->production_MWh = 0;
807     this->dispatch_MWh = 0;
808     this->dispatchable_MWh = 0;
809
810     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
811
812     this->capacity_factor_vec.resize(30, 0);
813     this->production_vec_MWh.resize(30, 0);
```

```
814     this->dispatch_vec_MWh.resize(30, 0);
815
816     this->tile_improvement_string = "SOLAR PV ARRAY";
817
818     this->__setUpTileImprovementSpriteStatic();
819     this->__computeCapacityFactors();
820     this->update();
821
822     std::cout « "SolarPV constructed at " « this « std::endl;
823
824     return;
825 }   /* SolarPV() */
```

4.11.2.2 ~SolarPV()

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

Destructor for the SolarPV class.

```
1165 {
1166     std::cout « "SolarPV at " « this « " destroyed" « std::endl;
1167
1168     return;
1169 }   /* ~SolarPV() */
```

4.11.3 Member Function Documentation

4.11.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
240 {
241     TileImprovement :: __breakdown();
242
243     this->production_MWh = 0;
244     this->dispatch_MWh = 0;
245     this->dispatchable_MWh = 0;
246     this->operation_maintenance_cost = 0;
247
248     return;
249 }   /* __breakdown() */
```

4.11.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors (by definition in the closed interval [0, 1]).

```
298 {
299     if (this->is_broken) {
300         return;
301     }
302 
303     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
304     std::default_random_engine generator(seed);
305 
306     double mean =
307         this->tile_resource_scalar * MEAN_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
308 
309     double stdev = STDEV_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
310 
311     if (this->tile_resource_scalar > 1) {
312         stdev /= this->tile_resource_scalar;
313     }
314 
315     double performance_factor = this->__getPerformanceFactor();
316 
317     std::normal_distribution<double> normal_dist(mean, stdev);
318 
319     double capacity_factor = 0;
320 
321     for (int i = 0; i < 30; i++) {
322         capacity_factor = performance_factor * normal_dist(generator);
323 
324         if (capacity_factor < 0) {
325             capacity_factor = 0;
326         }
327 
328         else if (capacity_factor > 1) {
329             capacity_factor = 1;
330         }
331 
332         this->capacity_factor_vec[i] = capacity_factor;
333     }
334 
335     return;
336 }   /* __computeCapacityFactors() */
```

4.11.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
384 {
385     if (this->is_broken) {
386         this->dispatchable_MWh = 0;
387         return;
388     }
389 
390     double stored_energy_MWh = 0;
391     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
392 
393     double demand_MWh = 0;
394     double production_MWh = 0;
395     double dispatchable_MWh = 0;
396     double difference_MWh = 0;
397 
398     double room_MWh = 0;
399 
400     for (int i = 0; i < 30; i++) {
401         demand_MWh = this->demand_vec_MWh[i];
402         production_MWh = this->production_vec_MWh[i];
403 
404         if (production_MWh <= demand_MWh) {
405             this->dispatch_vec_MWh[i] = production_MWh;
406             dispatchable_MWh += this->dispatch_vec_MWh[i];
```

```
407
408             difference_MWh = demand_MWh - production_MWh;
409
410             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
411                 if (difference_MWh > stored_energy_MWh) {
412                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
413                     dispatchable_MWh += stored_energy_MWh;
414                     stored_energy_MWh = 0;
415                 }
416
417                 else {
418                     this->dispatch_vec_MWh[i] += difference_MWh;
419                     dispatchable_MWh += difference_MWh;
420                     stored_energy_MWh -= difference_MWh;
421                 }
422             }
423         }
424
425         else {
426             this->dispatch_vec_MWh[i] = demand_MWh;
427             dispatchable_MWh += this->dispatch_vec_MWh[i];
428
429             difference_MWh = production_MWh - demand_MWh;
430
431             if (
432                 (storage_capacity_MWh > 0) and
433                 (stored_energy_MWh < storage_capacity_MWh)
434             ) {
435                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
436
437                 if (difference_MWh > room_MWh) {
438                     stored_energy_MWh += room_MWh;
439                 }
440
441                 else {
442                     stored_energy_MWh += difference_MWh;
443                 }
444             }
445         }
446     }
447
448     this->dispatchable_MWh = round(dispatchable_MWh);
449
450     if (this->dispatch_MWh <= 0) {
451         this->dispatch_MWh = 0;
452     }
453
454     else if (this->dispatch_MWh != this->dispatchable_MWh) {
455         this->dispatch_MWh = this->dispatchable_MWh;
456     }
457
458     return;
459 }   /* __computeDispatch() */
```

4.11.3.4 __computeProduction()

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

Helper method to compute production values.

```
351 {
352     if (this->is_broken) {
353         this->production_MWh = 0;
354         return;
355     }
356
357     double production_MWh = 0;
358
359     for (int i = 0; i < 30; i++) {
360         this->production_vec_MWh[i] =
361             this->max_daily_production_MWh * this->capacity_factor_vec[i];
362
363         production_MWh += this->production_vec_MWh[i];
364     }
365
366     this->production_MWh = round(production_MWh);
367
368     return;
369 }   /* __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     if (this->is_running) {
214         double operation_maintenance_cost =
215             (this->production_MWh * SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
216 
217         this->operation_maintenance_cost = round(operation_maintenance_cost);
218     }
219 
220     else {
221         this->operation_maintenance_cost = 0;
222     }
223 
224     return;
225 }   /* __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.

```
600 {
601     //  1. draw power capacity upgrade sprite
602     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
603     this->tile_improvement_sprite_static.setPosition(400 - 100, 400 - 32);
604 
605     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
606     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
607 
608     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();
609     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
610 
611     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
612 
613     this->tile_improvement_sprite_static.setPosition(initial_position);
614     this->tile_improvement_sprite_static.setColor(initial_colour);
615     this->tile_improvement_sprite_static.setScale(initial_scale);
616 
617     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
618 
619 
620     //  2. draw power capacity upgrade text
621     //                  16 char line = "                \n"
622     std::string power_upgrade_string = "POWER CAPACITY  \n";
623     power_upgrade_string            += "                \n";
624 
625     power_upgrade_string            += "CAPACITY:  ";
626     power_upgrade_string            += std::to_string(this->capacity_kW);
627     power_upgrade_string            += " kW\n";
628 
629     power_upgrade_string            += "LEVEL:     ";
630     power_upgrade_string            += std::to_string(this->upgrade_level);
631     power_upgrade_string            += "\n\n";
632 
633     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
634         power_upgrade_string        += "[W]: + 100 kW (";
635         power_upgrade_string        +=  std::to_string(SOLAR_PV_BUILD_COST);
636         power_upgrade_string        += " K)\n";
637     }
638 
639     else {
640         power_upgrade_string        += " * MAX LEVEL *  \n";
641     }
642 
643     sf::Text power_upgrade_text = sf::Text(
644         power_upgrade_string,
645         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
646         16
647     );
648 
649     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
650     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
651     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
652 
653     this->render_window_ptr->draw(power_upgrade_text);
654 
655 
656     //  3. draw energy capacity (storage) upgrade sprite
657     this->render_window_ptr->draw(this->storage_upgrade_sprite);
658     this->render_window_ptr->draw(this->upgrade_plus_sprite);
659 
660 
661     //  4. draw energy capacity (storage) upgrade text
662     //                    16 char line = "                \n"
663     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
664     energy_upgrade_string            += "                \n";
665 
666     energy_upgrade_string            += "CAPACITY:  ";
667     energy_upgrade_string            += std::to_string(this->storage_level * 200);
668     energy_upgrade_string            += " kWh\n";
669 
670     energy_upgrade_string            += "LEVEL:     ";
671     energy_upgrade_string            += std::to_string(this->storage_level);
672     energy_upgrade_string            += "\n\n";
673 
674     if (this->storage_level < MAX_STORAGE_LEVELS) {
675         energy_upgrade_string        += "[D]: + 200 kWh (";
676         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
677         energy_upgrade_string        += " K)\n";
```

```
678     }
679
680     else {
681         energy_upgrade_string += " * MAX LEVEL *  \n";
682     }
683
684     sf::Text energy_upgrade_text = sf::Text(
685         energy_upgrade_string,
686         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
687         16
688     );
689
690     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
691     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
692     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
693
694     this->render_window_ptr->draw(energy_upgrade_text);
695
696     return;
697 }   /* __drawUpgradeOptions() */
```

4.11.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
474 {
475     if (this->just_built) {
476         return;
477     }
478
479     switch (this->event_ptr->key.code) {
480         case (sf::Keyboard::U): {
481             this->__openUpgradeMenu();
482
483             break;
484         }
485
486
487         case (sf::Keyboard::W): {
488             if (this->production_menu_open) {
489                 this->dispatch_MWh++;
490
491                 if (this->dispatch_MWh > this->dispatchable_MWh) {
492                     this->dispatch_MWh = 0;
493                 }
494
495                 this->__computeProductionCosts();
496                 this->assets_manager_ptr->getSound("interface click")->play();
497             }
498
499             else if (this->upgrade_menu_open) {
500                 this->__upgradePowerCapacity();
501             }
502
503             break;
504         }
505
506
507         case (sf::Keyboard::S): {
508             if (this->production_menu_open) {
509                 this->dispatch_MWh--;
510
511                 if (this->dispatch_MWh < 0) {
512                     this->dispatch_MWh = this->dispatchable_MWh;
513                 }
514
515                 this->__computeProductionCosts();
516                 this->assets_manager_ptr->getSound("interface click")->play();
517             }
518
519             break;
520         }
521
522
523         case (sf::Keyboard::D): {
524             if (this->upgrade_menu_open) {
```

```
525                 this->__upgradeStorageCapacity();
526                 this->__computeProduction();
527                 this->__computeDispatch();
528             }
529 
530             break;
531         }
532 
533 
534         default: {
535             // do nothing!
536 
537             break;
538         }
539     }
540 
541     return;
542 }   /* __handleKeyPressEvents() */
```

4.11.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
557 {
558     if (this->just_built) {
559         return;
560     }
561 
562     switch (this->event_ptr->mouseButton.button) {
563         case (sf::Mouse::Left): {
564             //...
565 
566             break;
567         }
568 
569 
570         case (sf::Mouse::Right): {
571             //...
572 
573             break;
574         }
575 
576 
577         default: {
578             // do nothing!
579 
580             break;
581         }
582     }
583 
584     return;
585 }   /* __handleMouseButtonEvents() */
```

4.11.3.10 __repair()

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

Helper method to repair the solar PV array.

Reimplemented from TileImprovement.

```
264 {
265     if (this->credits < SOLAR_PV_BUILD_COST) {
266         std::cout « "Cannot repair solar PV: insufficient credits (need "
267             « SOLAR_PV_BUILD_COST « " K)" « std::endl;
268 
```

```
269         this->__sendInsufficientCreditsMessage();
270         return;
271     }
272
273     TileImprovement :: __repair();
274
275     this->just_upgraded = true;
276
277     this->__sendCreditsSpentMessage(SOLAR_PV_BUILD_COST);
278     this->__sendTileStateRequest();
279     this->__sendGameStateRequest();
280
281     return;
282 }   /* __repair() */
```

4.11.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
712 {
713     Message improvement_state_message;
714
715     improvement_state_message.channel = GAME_CHANNEL;
716     improvement_state_message.subject = "improvement state";
717
718     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
719     improvement_state_message.int_payload["operation_maintenance_cost"] =
720         this->operation_maintenance_cost;
721
722     this->message_hub_ptr->sendMessage(improvement_state_message);
723
724     std::cout « "Improvement state message sent by " « this « std::endl;
725
726     return;
727 }   /* __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.

```
930 {
931     //  1. send improvement state message
932     this->__sendImprovementStateMessage();
933 
934     //  2. update
935     this->__computeCapacityFactors();
936     this->update();
937 
938     //  3. handle start/stop
939     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
940         this->is_running = true;
941     }
942 
943     else if (this->is_running and (this->dispatch_MWh <= 0)) {
944         this->is_running = false;
945     }
946 
947     //  4. handle equipment health and breakdowns
948     if (this->is_running) {
949         this->health--;
950 
951         if (this->health <= 50) {
952             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
953 
954             if ((double)rand() / RAND_MAX <= breakdown_prob) {
955                 this->health = 0;
```

```
956             }
957         }
958 
959         if (this->health <= 0) {
960             this->__breakdown();
961         }
962     }
963 
964     //  5. send tile state request (if selected)
965     if (this->is_selected) {
966         this->__sendTileStateRequest();
967     }
968 
969     return;
970 }   /* 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.

```
1059 {
1060     //  1. if just built, call base method and return
1061     if (this->just_built) {
1062         TileImprovement :: draw();
1063 
1064         return;
1065     }
1066 
1067 
1068     //  2. handle upgrade effects
1069     if (this->just_upgraded) {
1070         this->tile_improvement_sprite_static.setColor(
1071             sf::Color(
1072                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1073                 255,
1074                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1075                 255
1076             )
1077         );
1078 
1079         this->tile_improvement_sprite_static.setScale(
1080             sf::Vector2f(
1081                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1082                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1083             )
1084         );
1085 
1086         this->upgrade_frame++;
1087     }
1088 
1089     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1090         this->tile_improvement_sprite_static.setColor(
1091             sf::Color(255,255,255,255)
1092         );
1093 
1094         this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
1095 
1096         this->just_upgraded = false;
1097         this->upgrade_frame = 0;
1098     }
1099 
1100 
1101     //  3. draw static sprite
1102     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
1103 
1104 
1105     //  4. draw storage upgrades
1106     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1107         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1108     }
1109 
1110 
1111     //  5. handle dispatch illustration
```

```
1112     if (this->dispatch_MWh > 0) {
1113         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1114         this->__drawDispatch();
1115     }
1116 
1117 
1118     //  6. draw production menu
1119     if (this->production_menu_open) {
1120         this->render_window_ptr->draw(this->production_menu_backing);
1121         this->render_window_ptr->draw(this->production_menu_backing_text);
1122 
1123         this->__drawProductionMenu();
1124     }
1125 
1126 
1127     //  7. draw upgrade menu
1128     if (this->upgrade_menu_open) {
1129         this->render_window_ptr->draw(this->upgrade_menu_backing);
1130         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1131 
1132         this->__drawUpgradeOptions();
1133     }
1134 
1135 
1136     //  10. handle broken effects
1137     if (this->is_broken) {
1138         this->tile_improvement_sprite_static.setColor(
1139             sf::Color(
1140                 255,
1141                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1142                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1143                 255
1144             )
1145         );
1146     }
1147 
1148     this->frame++;
1149     return;
1150 }   /* 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.

```
842 {
843     //                  32 char x 17 line console "--------------------------------\n";
844     std::string options_substring                = "CAPACITY:      ";
845     options_substring                           += std::to_string(this->capacity_kW);
846     options_substring                           += " kW (level ";
847     options_substring                           += std::to_string(this->upgrade_level);
848     options_substring                           += ")\n";
849 
850     options_substring                           += "PRODUCTION:    ";
851     options_substring                           += std::to_string(this->production_MWh);
852     options_substring                           += " MWh\n";
853 
854     options_substring                           += "DISPATCHABLE:  ";
855     options_substring                           += std::to_string(this->dispatchable_MWh);
856     options_substring                           += " MWh\n";
857 
858     options_substring                           += "HEALTH:      ";
859     options_substring                           += std::to_string(this->health);
860     options_substring                           += "/100";
861 
862     if (this->health <= 0) {
863         options_substring                       += " ** BROKEN! **\n";
```

```
864     }
865
866     else {
867         options_substring                       += "\n";
868     }
869
870     options_substring                           += "                                    \n";
871     options_substring                           += "   **** SOLAR PV OPTIONS ****   \n";
872     options_substring                           += "                                    \n";
873
874     if (this->is_broken) {
875         options_substring                       += "     [R]:  REPAIR (";
876         options_substring                       += std::to_string(SOLAR_PV_BUILD_COST);
877         options_substring                       += " K)\n";
878     }
879
880     else {
881         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
882     }
883
884     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
885     options_substring                           += "HOLD [P]:  SCRAP (";
886     options_substring                           += std::to_string(SCRAP_COST);
887     options_substring                           += " K)";
888
889     return options_substring;
890 }   /* getTileOptionsSubstring() */
```

4.11.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
1010 {
1011     TileImprovement :: processEvent();
1012
1013     if (this->event_ptr->type == sf::Event::KeyPressed) {
1014         this->__handleKeyPressEvents();
1015     }
1016
1017     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1018         this->__handleMouseButtonEvents();
1019     }
1020
1021     return;
1022 }   /* processEvent() */
```

4.11.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1037 {
1038     TileImprovement :: processMessage();
1039
1040     //...
1041
1042     return;
1043 }   /* 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.

```
907 {
908     TileImprovement :: setIsSelected(is_selected);
909 
910     if (this->is_running and this->is_selected) {
911         this->assets_manager_ptr->getSound("solar hum")->play();
912     }
913 
914     return;
915 }   /* setIsSelected() */
```

4.11.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
985 {
986     this->__computeProduction();
987     this->__computeProductionCosts();
988     this->__computeDispatch();
989 
990     if (this->is_selected) {
991         this->__sendTileStateRequest();
992     }
993 
994     return;
995 }   /* 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:

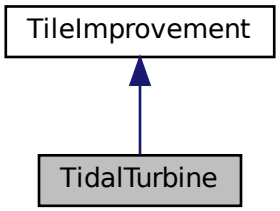
- header/SolarPV.h
- source/SolarPV.cpp

4.12 TidalTurbine Class Reference

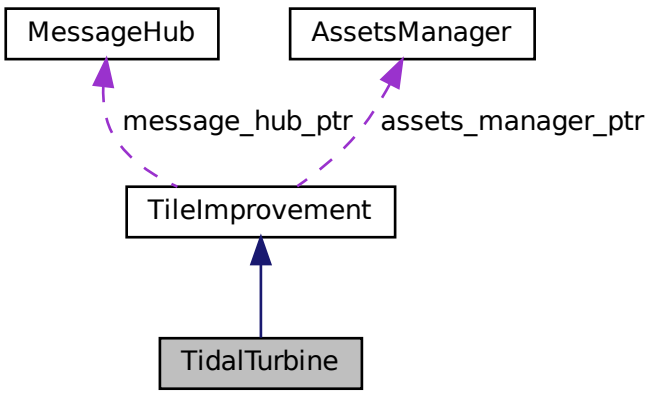
A settlement class (child class of TileImprovement).

```
#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 (by definition in the closed interval [0, 1]).
- 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.

```
775   :
776 TileImprovement(
777     position_x,
778     position_y,
779     tile_resource,
780     event_ptr,
781     render_window_ptr,
782     assets_manager_ptr,
783     message_hub_ptr
784 )
785 {
786     //  1. set attributes
787
788     //  1.1. private
789     //...
790
791     //  1.2. public
792     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
793
794     this->is_running = false;
795
796     this->health = 100;
797
798     this->capacity_kW = 100;
799     this->upgrade_level = 1;
800
801     this->storage_kWh = 0;
802     this->storage_level = 0;
803
804     this->production_MWh = 0;
805     this->dispatch_MWh = 0;
806     this->dispatchable_MWh = 0;
807
808     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
809
810     this->rotor_drotation = 64 * SECONDS_PER_FRAME;
811     this->bobbing_y = 4;
812
813     this->capacity_factor_vec.resize(30, 0);
814     this->production_vec_MWh.resize(30, 0);
815     this->dispatch_vec_MWh.resize(30, 0);
816
817     this->tile_improvement_string = "TIDAL TURBINE";
818
819     this->__setUpTileImprovementSpriteAnimated();
820     this->__computeCapacityFactors();
821     this->update();
822
823     std::cout « "TidalTurbine constructed at " « this « std::endl;
824
825     return;
826 }   /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

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

Destructor for the TidalTurbine class.

```
1192 {
1193     std::cout « "TidalTurbine at " « this « " destroyed" « std::endl;
1194
1195     return;
1196 }   /* ~TidalTurbine() */
```

4.12.3 Member Function Documentation

4.12.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
257 {
258     TileImprovement :: __breakdown();
259
260     this->production_MWh = 0;
261     this->dispatch_MWh = 0;
262     this->dispatchable_MWh = 0;
263     this->operation_maintenance_cost = 0;
264
265     return;
266 }   /* __breakdown() */
```

4.12.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors (by definition in the closed interval [0, 1]).

```
315 {
316     if (this->is_broken) {
317         return;
318     }
319
320     double performance_factor = this->__getPerformanceFactor();
321
322     for (int i = 0; i < 30; i++) {
323         this->capacity_factor_vec[i] =
324             performance_factor * this->tile_resource_scalar * DAILY_TIDAL_CAPACITY_FACTOR;
325     }
326
327     return;
328 }   /* __computeCapacityFactors() */
```

4.12.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
376 {
377     if (this->is_broken) {
378         this->dispatchable_MWh = 0;
379         return;
380     }
381
382     double stored_energy_MWh = 0;
383     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
384
385     double demand_MWh = 0;
386     double production_MWh = 0;
387     double dispatchable_MWh = 0;
388     double difference_MWh = 0;
```

```
389
390     double room_MWh = 0;
391
392     for (int i = 0; i < 30; i++) {
393         demand_MWh = this->demand_vec_MWh[i];
394         production_MWh = this->production_vec_MWh[i];
395
396         if (production_MWh <= demand_MWh) {
397             this->dispatch_vec_MWh[i] = production_MWh;
398             dispatchable_MWh += this->dispatch_vec_MWh[i];
399
400             difference_MWh = demand_MWh - production_MWh;
401
402             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
403                 if (difference_MWh > stored_energy_MWh) {
404                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
405                     dispatchable_MWh += stored_energy_MWh;
406                     stored_energy_MWh = 0;
407                 }
408
409                 else {
410                     this->dispatch_vec_MWh[i] += difference_MWh;
411                     dispatchable_MWh += difference_MWh;
412                     stored_energy_MWh -= difference_MWh;
413                 }
414             }
415         }
416
417         else {
418             this->dispatch_vec_MWh[i] = demand_MWh;
419             dispatchable_MWh += this->dispatch_vec_MWh[i];
420
421             difference_MWh = production_MWh - demand_MWh;
422
423             if (
424                 (storage_capacity_MWh > 0) and
425                 (stored_energy_MWh < storage_capacity_MWh)
426             ) {
427                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
428
429                 if (difference_MWh > room_MWh) {
430                     stored_energy_MWh += room_MWh;
431                 }
432
433                 else {
434                     stored_energy_MWh += difference_MWh;
435                 }
436             }
437         }
438     }
439
440     this->dispatchable_MWh = round(dispatchable_MWh);
441
442     if (this->dispatch_MWh <= 0) {
443         this->dispatch_MWh = 0;
444     }
445
446     else if (this->dispatch_MWh != this->dispatchable_MWh) {
447         this->dispatch_MWh = this->dispatchable_MWh;
448     }
449
450     return;
451 }   /* __computeDispatch() */
```

4.12.3.4 __computeProduction()

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

Helper method to compute production values.

```
343 {
344     if (this->is_broken) {
345         this->production_MWh = 0;
346         return;
347     }
348
349     double production_MWh = 0;
350
```

```
351     for (int i = 0; i < 30; i++) {
352         this->production_vec_MWh[i] =
353             this->max_daily_production_MWh * this->capacity_factor_vec[i];
354 
355         production_MWh += this->production_vec_MWh[i];
356     }
357 
358     this->production_MWh = round(production_MWh);
359 
360     return;
361 }   /* __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     if (this->is_running) {
231         double operation_maintenance_cost =
232             (this->production_MWh * TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
233 
234         this->operation_maintenance_cost = round(operation_maintenance_cost);
235     }
236 
237     else {
238         this->operation_maintenance_cost = 0;
239     }
240 
241     return;
242 }   /* __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.

```
592 {
593     //  1. draw power capacity upgrade sprite
594     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
595         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
596         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
597 
598         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
599         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
600 
601         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
602         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
603 
604         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
605         this->tile_improvement_sprite_animated[i].setRotation(0);
606 
607         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
608 
609         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
610         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
611         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
612         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
613     }
614 
615     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
616 
617 
618     //  2. draw power capacity upgrade text
619     //                  16 char line = "                \n"
620     std::string power_upgrade_string = "POWER CAPACITY  \n";
621     power_upgrade_string            += "                \n";
622 
623     power_upgrade_string            += "CAPACITY:  ";
624     power_upgrade_string            += std::to_string(this->capacity_kW);
625     power_upgrade_string            += " kW\n";
626 
627     power_upgrade_string            += "LEVEL:     ";
628     power_upgrade_string            += std::to_string(this->upgrade_level);
629     power_upgrade_string            += "\n\n";
630 
631     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
632         power_upgrade_string        += "[W]: + 100 kW (";
633         power_upgrade_string        +=  std::to_string(TIDAL_TURBINE_BUILD_COST);
634         power_upgrade_string        += " K)\n";
635     }
636 
637     else {
638         power_upgrade_string        += " * MAX LEVEL *  \n";
639     }
```

```
640
641     sf::Text power_upgrade_text = sf::Text(
642         power_upgrade_string,
643         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
644         16
645     );
646
647     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
648     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
649     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
650
651     this->render_window_ptr->draw(power_upgrade_text);
652
653
654     //  3. draw energy capacity (storage) upgrade sprite
655     this->render_window_ptr->draw(this->storage_upgrade_sprite);
656     this->render_window_ptr->draw(this->upgrade_plus_sprite);
657
658
659     //  4. draw energy capacity (storage) upgrade text
660     //                   16 char line = "                \n"
661     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
662     energy_upgrade_string            += "                \n";
663
664     energy_upgrade_string            += "CAPACITY:  ";
665     energy_upgrade_string            += std::to_string(this->storage_level * 200);
666     energy_upgrade_string            += " kWh\n";
667
668     energy_upgrade_string            += "LEVEL:     ";
669     energy_upgrade_string            += std::to_string(this->storage_level);
670     energy_upgrade_string            += "\n\n";
671
672     if (this->storage_level < MAX_STORAGE_LEVELS) {
673         energy_upgrade_string        += "[D]: + 200 kWh (";
674         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
675         energy_upgrade_string        += " K)\n";
676     }
677
678     else {
679         energy_upgrade_string += " * MAX LEVEL *  \n";
680     }
681
682     sf::Text energy_upgrade_text = sf::Text(
683         energy_upgrade_string,
684         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
685         16
686     );
687
688     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
689     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
690     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
691
692     this->render_window_ptr->draw(energy_upgrade_text);
693
694     return;
695 }   /* __drawUpgradeOptions() */
```

4.12.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
466 {
467     if (this->just_built) {
468         return;
469     }
470
471     switch (this->event_ptr->key.code) {
472         case (sf::Keyboard::U): {
473             this->__openUpgradeMenu();
474
475             break;
476         }
477
478
479         case (sf::Keyboard::W): {
480             if (this->production_menu_open) {
```

```
481                 this->dispatch_MWh++;
482 
483                 if (this->dispatch_MWh > this->dispatchable_MWh) {
484                     this->dispatch_MWh = 0;
485                 }
486 
487                 this->__computeProductionCosts();
488                 this->assets_manager_ptr->getSound("interface click")->play();
489             }
490 
491             else if (this->upgrade_menu_open) {
492                 this->__upgradePowerCapacity();
493             }
494 
495             break;
496         }
497 
498 
499         case (sf::Keyboard::S): {
500             if (this->production_menu_open) {
501                 this->dispatch_MWh--;
502 
503                 if (this->dispatch_MWh < 0) {
504                     this->dispatch_MWh = this->dispatchable_MWh;
505                 }
506 
507                 this->__computeProductionCosts();
508                 this->assets_manager_ptr->getSound("interface click")->play();
509             }
510 
511             break;
512         }
513 
514 
515         case (sf::Keyboard::D): {
516             if (this->upgrade_menu_open) {
517                 this->__upgradeStorageCapacity();
518                 this->__computeProduction();
519                 this->__computeDispatch();
520             }
521 
522             break;
523         }
524 
525 
526         default: {
527             // do nothing!
528 
529             break;
530         }
531     }
532 
533     return;
534 }   /* __handleKeyPressEvents() */
```

4.12.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
549 {
550     if (this->just_built) {
551         return;
552     }
553 
554     switch (this->event_ptr->mouseButton.button) {
555         case (sf::Mouse::Left): {
556             //...
557 
558             break;
559         }
560 
561 
562         case (sf::Mouse::Right): {
563             //...
564 
565             break;
```

```
566         }
567
568
569         default: {
570             // do nothing!
571
572             break;
573         }
574     }
575
576     return;
577 }   /* __handleMouseButtonEvents() */
```

4.12.3.10 __repair()

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

Helper method to repair the tidal turbine.

Reimplemented from TileImprovement.

```
281 {
282     if (this->credits < TIDAL_TURBINE_BUILD_COST) {
283         std::cout « "Cannot repair tidal turbine: insufficient credits (need "
284             « TIDAL_TURBINE_BUILD_COST « " K)" « std::endl;
285
286         this->__sendInsufficientCreditsMessage();
287         return;
288     }
289
290     TileImprovement :: __repair();
291
292     this->just_upgraded = true;
293
294     this->__sendCreditsSpentMessage(TIDAL_TURBINE_BUILD_COST);
295     this->__sendTileStateRequest();
296     this->__sendGameStateRequest();
297
298     return;
299 }   /* __repair() */
```

4.12.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
710 {
711     Message improvement_state_message;
712
713     improvement_state_message.channel = GAME_CHANNEL;
714     improvement_state_message.subject = "improvement state";
715
716     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
717     improvement_state_message.int_payload["operation_maintenance_cost"] =
718         this->operation_maintenance_cost;
719
720     this->message_hub_ptr->sendMessage(improvement_state_message);
721
722     std::cout « "Improvement state message sent by " « this « std::endl;
723
724     return;
725 }   /* __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.

```
932 {
933     //  1. send improvement state message
934     this->__sendImprovementStateMessage();
935 
936     //  2. update
937     this->__computeCapacityFactors();
938     this->update();
939 
940     //  3. handle start/stop
941     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
942         this->is_running = true;
943     }
944 
945     else if (this->is_running and (this->dispatch_MWh <= 0)) {
946         this->is_running = false;
947     }
948 
949     //  4. handle equipment health and breakdowns
950     if (this->is_running) {
951         this->health--;
952 
953         if (this->health <= 50) {
954             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
955 
956             if ((double)rand() / RAND_MAX <= breakdown_prob) {
957                 this->health = 0;
958             }
959         }
960 
961         if (this->health <= 0) {
962             this->__breakdown();
963         }
964     }
965 
966     //  5. send tile state request (if selected)
967     if (this->is_selected) {
968         this->__sendTileStateRequest();
969     }
970 
971     return;
972 }   /* 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.

```
1061 {
1062     //  1. if just built, call base method and return
1063     if (this->just_built) {
1064         TileImprovement :: draw();
1065 
1066         return;
1067     }
1068 
1069 
1070     //  2. handle upgrade effects
1071     if (this->just_upgraded) {
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(
1075                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
```

```
1076                    255,
1077                    255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1078                    255
1079                )
1080            );
1081
1082            this->tile_improvement_sprite_animated[i].setScale(
1083                sf::Vector2f(
1084                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1085                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1086                )
1087            );
1088        }
1089
1090        this->upgrade_frame++;
1091    }
1092
1093    if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1094        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1095            this->tile_improvement_sprite_animated[i].setColor(
1096                sf::Color(255,255,255,255)
1097            );
1098
1099            this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1100        }
1101
1102        this->just_upgraded = false;
1103        this->upgrade_frame = 0;
1104    }
1105
1106
1107    //  3. handle bobbing
1108    for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1109        this->tile_improvement_sprite_animated[i].setPosition(
1110            this->position_x,
1111            this->position_y + this->bobbing_y * cos(
1112                (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1113            )
1114        );
1115    }
1116
1117
1118    //  4. draw first element of animated sprite
1119    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1120
1121
1122    //  5. draw second element of animated sprite
1123    if (this->is_running) {
1124        this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1125    }
1126
1127    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1128
1129
1130    //  6. draw storage upgrades
1131    for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1132        this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1133    }
1134
1135
1136    //  7. handle dispatch illustration
1137    if (this->dispatch_MWh > 0) {
1138        this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1139        this->__drawDispatch();
1140    }
1141
1142
1143    //  8. draw production menu
1144    if (this->production_menu_open) {
1145        this->render_window_ptr->draw(this->production_menu_backing);
1146        this->render_window_ptr->draw(this->production_menu_backing_text);
1147
1148        this->__drawProductionMenu();
1149    }
1150
1151
1152    //  9. draw upgrade menu
1153    if (this->upgrade_menu_open) {
1154        this->render_window_ptr->draw(this->upgrade_menu_backing);
1155        this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1156
1157        this->__drawUpgradeOptions();
1158    }
1159
1160
1161    //  10. handle broken effects
1162    if (this->is_broken) {
```

```
1163         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1164             this->tile_improvement_sprite_animated[i].setColor(
1165                 sf::Color(
1166                     255,
1167                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1168                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1169                     255
1170                 )
1171             );
1172         }
1173     }
1174
1175     this->frame++;
1176     return;
1177 }   /* 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.

```
843 {
844     //                  32 char x 17 line console "--------------------------------\n";
845     std::string options_substring                = "CAPACITY:      ";
846     options_substring                           += std::to_string(this->capacity_kW);
847     options_substring                           += " kW (level ";
848     options_substring                           += std::to_string(this->upgrade_level);
849     options_substring                           += ")\n";
850
851     options_substring                           += "PRODUCTION:    ";
852     options_substring                           += std::to_string(this->production_MWh);
853     options_substring                           += " MWh\n";
854
855     options_substring                           += "DISPATCHABLE:  ";
856     options_substring                           += std::to_string(this->dispatchable_MWh);
857     options_substring                           += " MWh\n";
858
859     options_substring                           += "HEALTH:      ";
860     options_substring                           += std::to_string(this->health);
861     options_substring                           += "/100";
862
863     if (this->health <= 0) {
864         options_substring                       += " ** BROKEN! **\n";
865     }
866
867     else {
868         options_substring                       += "\n";
869     }
870
871     options_substring                           += "                                \n";
872     options_substring                           += "**** TIDAL TURBINE OPTIONS **** \n";
873     options_substring                           += "                                \n";
874
875     if (this->is_broken) {
876         options_substring                       += "     [R]:  REPAIR (";
877         options_substring                       += std::to_string(TIDAL_TURBINE_BUILD_COST);
878         options_substring                       += " K)\n";
879     }
880
881     else {
882         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
883     }
884
885     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
886     options_substring                           += "HOLD [P]:  SCRAP (";
887     options_substring                           += std::to_string(SCRAP_COST);
888     options_substring                           += " K)";
889
890     return options_substring;
891 }   /* getTileOptionsSubstring() */
```

4.12.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
1012 {
1013     TileImprovement :: processEvent();
1014
1015     if (this->event_ptr->type == sf::Event::KeyPressed) {
1016         this->__handleKeyPressEvents();
1017     }
1018
1019     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1020         this->__handleMouseButtonEvents();
1021     }
1022
1023     return;
1024 }   /* processEvent() */
```

4.12.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1039 {
1040     TileImprovement :: processMessage();
1041
1042     //...
1043
1044     return;
1045 }   /* 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.

```
908 {
909     TileImprovement :: setIsSelected(is_selected);
910
911     if (this->is_running and this->is_selected) {
912         this->assets_manager_ptr->getSound("water flow")->play();
913     }
914
```

```
915     return;
916 }   /* setIsSelected() */
```

4.12.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
987 {
988     this->__computeProduction();
989     this->__computeProductionCosts();
990     this->__computeDispatch();
991 
992     if (this->is_selected) {
993         this->__sendTileStateRequest();
994     }
995 
996     return;
997 }   /* 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:

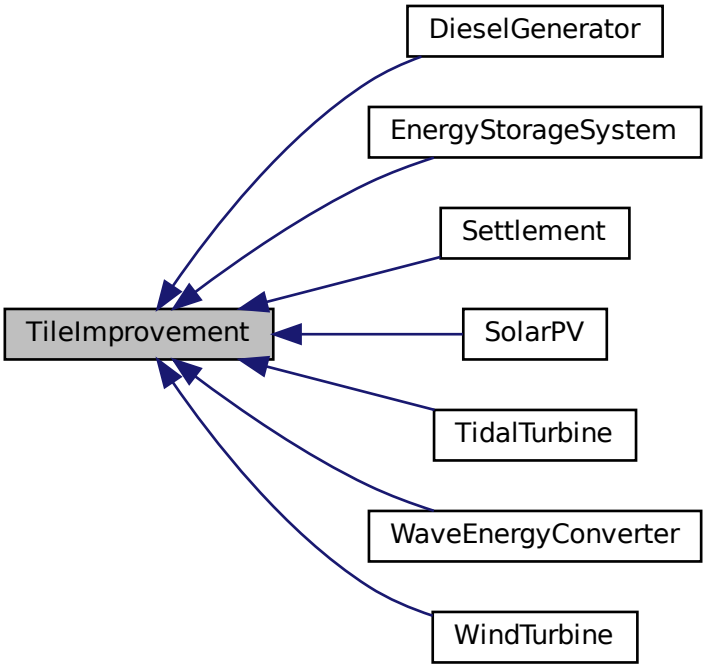
- header/TidalTurbine.h
- source/TidalTurbine.cpp

4.13 TileImprovement Class Reference

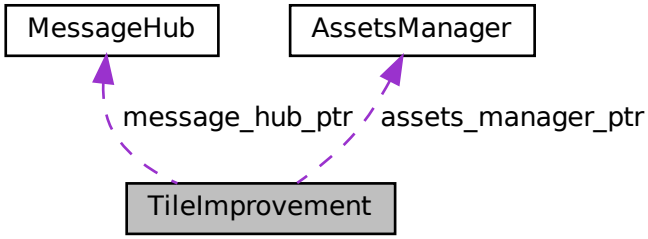
A base class for the tile improvement hierarchy.

```
#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.
- double __getPerformanceFactor (void)

  Helper method to compute and return performance factor as a function of state of health. For renewable assets, it affects production, whereas for diesel it effects fuel consumption.
- 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.

```
759 {
760     //  1. set attributes
761
762     //  1.1. protected
763     this->event_ptr = event_ptr;
764     this->render_window_ptr = render_window_ptr;
765
```

```
766     this->assets_manager_ptr = assets_manager_ptr;
767     this->message_hub_ptr = message_hub_ptr;
768 
769     //  1.2. public
770     this->is_selected = true;
771     this->just_built = true;
772     this->production_menu_open = false;
773     this->upgrade_menu_open = false;
774     this->is_broken = false;
775 
776     this->just_upgraded = false;
777     this->upgrade_frame = 0;
778 
779     this->frame = 0;
780     this->credits = 0;
781     this->month = 1;
782     this->demand_MWh = 0;
783 
784     this->demand_vec_MWh.resize(30, 0);
785 
786     this->operation_maintenance_cost = 0;
787 
788     this->tile_resource = tile_resource;
789 
790     switch (this->tile_resource) {
791         case (0): {
792             this->tile_resource_scalar = 0.85;
793 
794             break;
795         }
796 
797 
798         case (1): {
799             this->tile_resource_scalar = 0.925;
800 
801             break;
802         }
803 
804 
805         case (2): {
806             this->tile_resource_scalar = 1;
807 
808             break;
809         }
810 
811 
812         case (3): {
813             this->tile_resource_scalar = 1.075;
814 
815             break;
816         }
817 
818 
819         case (4): {
820             this->tile_resource_scalar = 1.15;
821 
822             break;
823         }
824 
825 
826         default: {
827             this->tile_resource_scalar = 1;
828         }
829     }
830 
831     this->position_x = position_x;
832     this->position_y = position_y;
833 
834     this->game_phase = "build settlement";
835 
836     this->__setUpProductionMenu();
837     this->__setUpUpgradeMenu();
838     this->__setUpDispatchIllustration();
839 
840     std::cout « "TileImprovement constructed at " « this « std::endl;
841 
842     return;
843 }   /* TileImprovement() */
```

4.13.2.2 ~TileImprovement()

```
TileImprovement::~TileImprovement (
```

```
            void  )  [virtual]
```

Destructor for the TileImprovement class.

```
1076 {
1077     std::cout « "TileImprovement at " « this « " destroyed" « std::endl;
1078
1079     return;
1080 }   /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
463 {
464     this->is_broken = true;
465     this->is_running = false;
466     this->update();
467     this->assets_manager_ptr->getSound("breakdown")->play();
468
469     return;
470 }   /* __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.

```
549 {
550     if (not this->upgrade_menu_open) {
551         return;
552     }
553
554     this->upgrade_menu_open = false;
555     this->assets_manager_ptr->getSound("build menu close")->play();
556
557     return;
558 }   /* __closeUpgradeMenu() */
```

4.13.3.4 __drawDispatch()

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

Helper method to draw dispatch illustration.

```
680 {
681     double alpha = 255 * pow(cos((0.5 * M_PI * this->frame) / FRAMES_PER_SECOND), 2);
682 
683 
684     //  1. dispatch backing
685     sf::Color backing_colour = this->dispatch_backing.getFillColor();
686     backing_colour.a = alpha;
687 
688     this->dispatch_backing.setFillColor(backing_colour);
689     this->dispatch_backing.setOutlineColor(sf::Color(0, 0, 0, alpha));
690 
691     this->render_window_ptr->draw(this->dispatch_backing);
692 
693 
694     //  2. dispatch text
695     this->dispatch_text.setOrigin(
696         this->dispatch_text.getLocalBounds().width / 2,
697         this->dispatch_text.getLocalBounds().height / 2
698     );
699 
700     sf::Color text_colour = this->dispatch_text.getFillColor();
701     text_colour.a = alpha;
702 
703     this->dispatch_text.setFillColor(text_colour);
704 
705     this->render_window_ptr->draw(this->dispatch_text);
706 
707     return;
708 }   /* __drawDispatch() */
```

4.13.3.5 __getPerformanceFactor()

```
double TileImprovement::__getPerformanceFactor (
            void  )  [protected]
```

Helper method to compute and return performance factor as a function of state of health. For renewable assets, it affects production, whereas for diesel it effects fuel consumption.

Returns

A performance factor (in the close interval[0, 1]).

```
435 {
436     double performance_factor =
437         PERFORMANCE_FACTOR_COEFFICIENT * pow(this->health, PERFORMANCE_FACTOR_EXPONENT);
438 
439     if (performance_factor < 0) {
440         performance_factor = 0;
441     }
442 
443     else if (performance_factor > 1) {
444         performance_factor = 1;
445     }
446 
447     return performance_factor;
448 }   /* __getPerformanceFactor(void) */
```

4.13.3.6 __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() */
```

4.13.3.7 __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() */
```

4.13.3.8 __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() */
```

4.13.3.9 __openUpgradeMenu()

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

Helper method to open the upgrade menu.

```
521 {
522     if (this->upgrade_menu_open) {
523         return;
524     }
525
526     if (this->production_menu_open) {
527         this->__closeProductionMenu();
528     }
529
530     this->upgrade_menu_open = true;
531     this->assets_manager_ptr->getSound("build menu open")->play();
532
533     return;
534 }   /* __openUpgradeMenu() */
```

4.13.3.10 __repair()

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

Helper method to repair a tile improvement.

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

```
485 {
486     this->health = 100;
487     if (this->is_broken) {
488         this->is_broken = false;
489         this->assets_manager_ptr->getSound("positive notification")->play();
490     }
491 
492 
493     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
494         this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
495     }
496 
497     else {
498         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
499             this->tile_improvement_sprite_animated[i].setColor(
500                 sf::Color(255, 255, 255, 255)
501             );
502         }
503     }
504 
505     return;
506 }   /* __repair() */
```

4.13.3.11 __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.

```
626 {
627     Message credits_spent_message;
628 
629     credits_spent_message.channel = GAME_CHANNEL;
630     credits_spent_message.subject = "credits spent";
631 
632     credits_spent_message.int_payload["credits spent"] = credits_spent;
633 
634     this->message_hub_ptr->sendMessage(credits_spent_message);
635 
636     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
637         << std::endl;
638     return;
639 }   /* __sendCreditsSpentMessage() */
```

4.13.3.12 __sendGameStateRequest()

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

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

```
599 {
600     Message game_state_request;
601 
602     game_state_request.channel = GAME_CHANNEL;
603     game_state_request.subject = "state request";
604 
605     this->message_hub_ptr->sendMessage(game_state_request);
606 
607     std::cout << "Game state request message sent by " << this << std::endl;
608     return;
609 }   /* __sendGameStateRequest() */
```

4.13.3.13 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
654 {
655     Message insufficient_credits_message;
656 
657     insufficient_credits_message.channel = GAME_CHANNEL;
658     insufficient_credits_message.subject = "insufficient credits";
659 
660     this->message_hub_ptr->sendMessage(insufficient_credits_message);
661 
662     std::cout << "Insufficient credits message sent by " << this << std::endl;
663 
664     return;
665 }   /* __sendInsufficientCreditsMessage() */
```

4.13.3.14 __sendTileStateRequest()

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

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

```
574 {
575     Message tile_state_request;
576 
577     tile_state_request.channel = TILE_STATE_CHANNEL;
578     tile_state_request.subject = "state request";
579 
580     this->message_hub_ptr->sendMessage(tile_state_request);
581 
582     std::cout << "Tile state request sent by " << this << std::endl;
583     return;
584 }   /* __sendTileStateRequest() */
```

4.13.3.15 __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() */
```

4.13.3.16 __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() */
```

4.13.3.17 __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() */
```

4.13.3.18 __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() */
```

4.13.3.19 advanceTurn()

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

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

```
192 {return;}
```

4.13.3.20 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.

```
947 {
948     if (this->tile_improvement_sprite_static.getTexture() != NULL) {
949         int alpha = this->tile_improvement_sprite_static.getColor().a;
950 
951         alpha += 0.08 * FRAMES_PER_SECOND;
952 
953         this->tile_improvement_sprite_static.setColor(
954             sf::Color(255, 255, 255, alpha)
955         );
956 
957         this->tile_improvement_sprite_static.move(0, 50 * SECONDS_PER_FRAME);
```

```
958
959         if (
960             (alpha >= 255) or
961             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
962         ) {
963             this->tile_improvement_sprite_static.setColor(
964                 sf::Color(255, 255, 255, 255)
965             );
966
967             this->tile_improvement_sprite_static.setPosition(
968                 this->position_x,
969                 this->position_y + 12
970             );
971
972             this->just_built = false;
973             this->assets_manager_ptr->getSound("place improvement")->play();
974         }
975
976         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
977     }
978
979
980     else {
981         int alpha = 0;
982
983         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
984             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
985
986             alpha += 0.08 * FRAMES_PER_SECOND;
987
988             this->tile_improvement_sprite_animated[i].setColor(
989                 sf::Color(255, 255, 255, alpha)
990             );
991
992             this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
993
994             if (
995                 (alpha >= 255) or
996                 (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
997             ) {
998                 this->tile_improvement_sprite_animated[i].setColor(
999                     sf::Color(255, 255, 255, 255)
1000                 );
1001
1002                 this->tile_improvement_sprite_animated[i].setPosition(
1003                     this->position_x,
1004                     this->position_y + 12
1005                 );
1006             }
1007
1008             this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
1009         }
1010
1011         if (
1012             (alpha >= 255) or
1013             (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
1014         ) {
1015             this->just_built = false;
1016             this->assets_manager_ptr->getSound("place improvement")->play();
1017
1018             switch (this->tile_improvement_type) {
1019                 case (TileImprovementType :: WIND_TURBINE): {
1020                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1021                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
1022                         this->tile_improvement_sprite_animated[i].move(0, -32);
1023                     }
1024
1025                     break;
1026                 }
1027
1028
1029                 case (TileImprovementType :: TIDAL_TURBINE): {
1030                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1031                         this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
1032                         this->tile_improvement_sprite_animated[i].move(0, -19);
1033                     }
1034
1035                     break;
1036                 }
1037
1038
1039                 case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
1040                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1041                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
1042                         this->tile_improvement_sprite_animated[i].move(0, -32);
1043                     }
1044
```

```
1045                     break;
1046                 }
1047
1048
1049                 default: {
1050                     // do nothing!
1051
1052                     break;
1053                 }
1054             }
1055         }
1056     }
1057
1058
1059     this->frame++;
1060     return;
1061 }   /* draw() */
```

4.13.3.21 getTileOptionsSubstring()

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

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

```
196 {return "";}
```

4.13.3.22 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.

```
887 {
888     if (this->event_ptr->type == sf::Event::KeyPressed) {
889         this->__handleKeyPressEvents();
890     }
891
892     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
893         this->__handleMouseButtonEvents();
894     }
895
896     return;
897 }   /* processEvent() */
```

4.13.3.23 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.

```
912 {
913     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
914         Message game_state_message = this->message_hub_ptr->receiveMessage(
915             GAME_STATE_CHANNEL
916         );
917 
918         if (game_state_message.subject == "turn advance") {
919             this->credits = game_state_message.int_payload["credits"];
920             this->month = game_state_message.int_payload["month"];
921             this->demand_MWh = game_state_message.int_payload["demand_MWh"];
922 
923             this->advanceTurn();
924 
925             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
926             std::cout << "Turn advance message read and passed by " << this << std::endl;
927         }
928     }
929 
930     return;
931 }   /* processMessage() */
```

4.13.3.24 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.

```
860 {
861     this->is_selected = is_selected;
862 
863     if ((not is_selected) and this->production_menu_open) {
864         this->__closeProductionMenu();
865     }
866 
867     if ((not is_selected) and this->upgrade_menu_open) {
868         this->__closeUpgradeMenu();
869     }
870 
871     return;
872 }   /* setIsSelected() */
```

4.13.3.25 update()

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, and SolarPV.
194 {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:

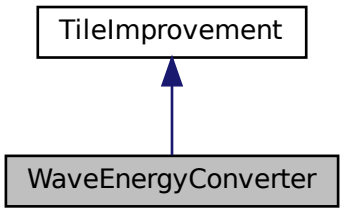
- header/TileImprovement.h
- source/TileImprovement.cpp

4.14 WaveEnergyConverter Class Reference

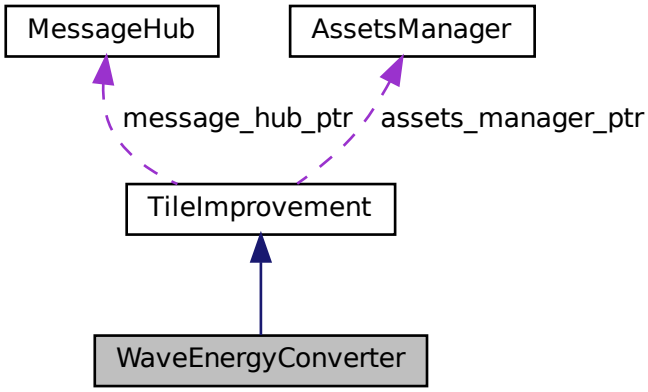
A settlement class (child class of TileImprovement).

`#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 (by definition in the closed interval [0, 1]).

- 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.

```
795   :
796 TileImprovement(
797     position_x,
798     position_y,
799     tile_resource,
800     event_ptr,
801     render_window_ptr,
802     assets_manager_ptr,
803     message_hub_ptr
804 )
805 {
806     //  1. set attributes
807
808     //  1.1. private
809     //...
810
811     //  1.2. public
812     this->tile_improvement_type = TileImprovementType :: WAVE_ENERGY_CONVERTER;
813
814     this->is_running = false;
815
816     this->health = 100;
817
818     this->capacity_kW = 100;
819     this->upgrade_level = 1;
820
821     this->storage_kWh = 0;
822     this->storage_level = 0;
823
824     this->production_MWh = 0;
825     this->dispatch_MWh = 0;
826     this->dispatchable_MWh = 0;
827
828     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
829
830     this->bobbing_y = 4;
831
832     this->capacity_factor_vec.resize(30, 0);
833     this->production_vec_MWh.resize(30, 0);
834     this->dispatch_vec_MWh.resize(30, 0);
835
836     this->tile_improvement_string = "WAVE ENERGY";
837
838     this->__setUpTileImprovementSpriteAnimated();
839     this->__computeCapacityFactors();
840     this->update();
841
842     std::cout « "WaveEnergyConverter constructed at " « this « std::endl;
843
844     return;
845 }   /* WaveEnergyConverter() */
```

4.14.2.2 ~WaveEnergyConverter()

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

Destructor for the WaveEnergyConverter class.

```
1222 {
1223     std::cout « "WaveEnergyConverter at " « this « " destroyed" « std::endl;
1224
1225     return;
1226 }   /* ~WaveEnergyConverter() */
```

4.14.3 Member Function Documentation

4.14.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
257 {
258     TileImprovement :: __breakdown();
259 
260     this->production_MWh = 0;
261     this->dispatch_MWh = 0;
262     this->dispatchable_MWh = 0;
263     this->operation_maintenance_cost = 0;
264 
265     return;
266 }   /* __breakdown() */
```

4.14.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors (by definition in the closed interval [0, 1]).

```
315 {
316     if (this->is_broken) {
317         return;
318     }
319 
320     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
321     std::default_random_engine generator(seed);
322 
323     double mean =
324         this->tile_resource_scalar * MEAN_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
325 
326     double stdev = STDEV_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
327 
328     if (this->tile_resource_scalar > 1) {
329         stdev /= this->tile_resource_scalar;
330     }
331 
332     double performance_factor = this->__getPerformanceFactor();
333 
334     std::normal_distribution<double> normal_dist(mean, stdev);
335 
336     double capacity_factor = 0;
337 
338     for (int i = 0; i < 30; i++) {
339         capacity_factor = performance_factor * normal_dist(generator);
340 
341         if (capacity_factor < 0) {
342             capacity_factor = 0;
343         }
344 
345         else if (capacity_factor > 1) {
346             capacity_factor = 1;
347         }
348 
349         this->capacity_factor_vec[i] = capacity_factor;
350     }
351 
352     return;
353 }   /* __computeCapacityFactors() */
```

4.14.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
401 {
402     if (this->is_broken) {
403         this->dispatchable_MWh = 0;
404         return;
405     }
406 
407     double stored_energy_MWh = 0;
408     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
409 
410     double demand_MWh = 0;
411     double production_MWh = 0;
412     double dispatchable_MWh = 0;
413     double difference_MWh = 0;
414 
415     double room_MWh = 0;
416 
417     for (int i = 0; i < 30; i++) {
418         demand_MWh = this->demand_vec_MWh[i];
419         production_MWh = this->production_vec_MWh[i];
420 
421         if (production_MWh <= demand_MWh) {
422             this->dispatch_vec_MWh[i] = production_MWh;
423             dispatchable_MWh += this->dispatch_vec_MWh[i];
424 
425             difference_MWh = demand_MWh - production_MWh;
426 
427             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
428                 if (difference_MWh > stored_energy_MWh) {
429                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
430                     dispatchable_MWh += stored_energy_MWh;
431                     stored_energy_MWh = 0;
432                 }
433 
434                 else {
435                     this->dispatch_vec_MWh[i] += difference_MWh;
436                     dispatchable_MWh += difference_MWh;
437                     stored_energy_MWh -= difference_MWh;
438                 }
439             }
440         }
441 
442         else {
443             this->dispatch_vec_MWh[i] = demand_MWh;
444             dispatchable_MWh += this->dispatch_vec_MWh[i];
445 
446             difference_MWh = production_MWh - demand_MWh;
447 
448             if (
449                 (storage_capacity_MWh > 0) and
450                 (stored_energy_MWh < storage_capacity_MWh)
451             ) {
452                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
453 
454                 if (difference_MWh > room_MWh) {
455                     stored_energy_MWh += room_MWh;
456                 }
457 
458                 else {
459                     stored_energy_MWh += difference_MWh;
460                 }
461             }
462         }
463     }
464 
465     this->dispatchable_MWh = round(dispatchable_MWh);
466 
467     if (this->dispatch_MWh <= 0) {
468         this->dispatch_MWh = 0;
469     }
470 
471     else if (this->dispatch_MWh != this->dispatchable_MWh) {
472         this->dispatch_MWh = this->dispatchable_MWh;
473     }
474 
475     return;
476 }   /* __computeDispatch() */
```

4.14.3.4 __computeProduction()

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

Helper method to compute production values.

```
368 {
369     if (this->is_broken) {
370         this->production_MWh = 0;
371         return;
372     }
373
374     double production_MWh = 0;
375
376     for (int i = 0; i < 30; i++) {
377         this->production_vec_MWh[i] =
378             this->max_daily_production_MWh * this->capacity_factor_vec[i];
379
380         production_MWh += this->production_vec_MWh[i];
381     }
382
383     this->production_MWh = round(production_MWh);
384
385     return;
386 }   /* __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     if (this->is_running) {
231         double operation_maintenance_cost =
232             (this->production_MWh * WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
233
234         this->operation_maintenance_cost = round(operation_maintenance_cost);
235     }
236
237     else {
238         this->operation_maintenance_cost = 0;
239     }
240
241     return;
242 }   /* __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.

```
616 {
617     //  1. draw power capacity upgrade sprite
618     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
619         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
620         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
621
622         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
623         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
624
625         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
626         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
627
628         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
629
630         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
631         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
632         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
633     }
634
635     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
636
637
638     //  2. draw power capacity upgrade text
639     //                  16 char line = "                \n"
640     std::string power_upgrade_string = "POWER CAPACITY  \n";
641     power_upgrade_string            += "                \n";
642
643     power_upgrade_string            += "CAPACITY:  ";
644     power_upgrade_string            += std::to_string(this->capacity_kW);
645     power_upgrade_string            += " kW\n";
646
```

```
647     power_upgrade_string            += "LEVEL:     ";
648     power_upgrade_string            += std::to_string(this->upgrade_level);
649     power_upgrade_string            += "\n\n";
650 
651     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
652         power_upgrade_string        += "[W]: + 100 kW (";
653         power_upgrade_string        +=  std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
654         power_upgrade_string        += " K)\n";
655     }
656 
657     else {
658         power_upgrade_string        += " * MAX LEVEL *  \n";
659     }
660 
661     sf::Text power_upgrade_text = sf::Text(
662         power_upgrade_string,
663         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
664         16
665     );
666 
667     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
668     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
669     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
670 
671     this->render_window_ptr->draw(power_upgrade_text);
672 
673 
674     //  3. draw energy capacity (storage) upgrade sprite
675     this->render_window_ptr->draw(this->storage_upgrade_sprite);
676     this->render_window_ptr->draw(this->upgrade_plus_sprite);
677 
678 
679     //  4. draw energy capacity (storage) upgrade text
680     //                  16 char line = "                \n"
681     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
682     energy_upgrade_string           += "                \n";
683 
684     energy_upgrade_string           += "CAPACITY:  ";
685     energy_upgrade_string           += std::to_string(this->storage_level * 200);
686     energy_upgrade_string           += " kWh\n";
687 
688     energy_upgrade_string           += "LEVEL:     ";
689     energy_upgrade_string           += std::to_string(this->storage_level);
690     energy_upgrade_string           += "\n\n";
691 
692     if (this->storage_level < MAX_STORAGE_LEVELS) {
693         energy_upgrade_string       += "[D]: + 200 kWh (";
694         energy_upgrade_string       +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
695         energy_upgrade_string       += " K)\n";
696     }
697 
698     else {
699         energy_upgrade_string += " * MAX LEVEL *  \n";
700     }
701 
702     sf::Text energy_upgrade_text = sf::Text(
703         energy_upgrade_string,
704         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
705         16
706     );
707 
708     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
709     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
710     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
711 
712     this->render_window_ptr->draw(energy_upgrade_text);
713 
714     return;
715 }   /* __drawUpgradeOptions() */
```

4.14.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
491 {
492     if (this->just_built) {
```

```
493         return;
494     }
495 
496     switch (this->event_ptr->key.code) {
497         case (sf::Keyboard::U): {
498             this->__openUpgradeMenu();
499 
500             break;
501         }
502 
503 
504         case (sf::Keyboard::W): {
505             if (this->production_menu_open) {
506                 this->dispatch_MWh++;
507 
508                 if (this->dispatch_MWh > this->dispatchable_MWh) {
509                     this->dispatch_MWh = 0;
510                 }
511 
512                 this->__computeProductionCosts();
513                 this->assets_manager_ptr->getSound("interface click")->play();
514             }
515 
516             else if (this->upgrade_menu_open) {
517                 this->__upgradePowerCapacity();
518             }
519 
520             break;
521         }
522 
523 
524         case (sf::Keyboard::S): {
525             if (this->production_menu_open) {
526                 this->dispatch_MWh--;
527 
528                 if (this->dispatch_MWh < 0) {
529                     this->dispatch_MWh = this->dispatchable_MWh;
530                 }
531 
532                 this->__computeProductionCosts();
533                 this->assets_manager_ptr->getSound("interface click")->play();
534             }
535 
536             break;
537         }
538 
539 
540         case (sf::Keyboard::D): {
541             if (this->upgrade_menu_open) {
542                 this->__upgradeStorageCapacity();
543                 this->__computeProduction();
544                 this->__computeDispatch();
545             }
546 
547             break;
548         }
549 
550 
551         default: {
552             // do nothing!
553 
554             break;
555         }
556     }
557 
558     return;
559 }   /* __handleKeyPressEvents() */
```

4.14.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
574 {
575     if (this->just_built) {
576         return;
577     }
```

```
578     switch (this->event_ptr->mouseButton.button) {
579         case (sf::Mouse::Left): {
580             //...
581 
582             break;
583         }
584 
585 
586         case (sf::Mouse::Right): {
587             //...
588 
589             break;
590         }
591 
592 
593         default: {
594             // do nothing!
595 
596             break;
597         }
598     }
599 
600     return;
601 }   /* __handleMouseButtonEvents() */
```

4.14.3.10 __repair()

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

Helper method to repair the wave energy converter.

Reimplemented from TileImprovement.

```
281 {
282     if (this->credits < WAVE_ENERGY_CONVERTER_BUILD_COST) {
283         std::cout « "Cannot repair wave energy converter: insufficient credits (need "
284             « WAVE_ENERGY_CONVERTER_BUILD_COST « " K)" « std::endl;
285 
286         this->__sendInsufficientCreditsMessage();
287         return;
288     }
289 
290     TileImprovement :: __repair();
291 
292     this->just_upgraded = true;
293 
294     this->__sendCreditsSpentMessage(WAVE_ENERGY_CONVERTER_BUILD_COST);
295     this->__sendTileStateRequest();
296     this->__sendGameStateRequest();
297 
298     return;
299 }   /* __repair() */
```

4.14.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
730 {
731     Message improvement_state_message;
732 
733     improvement_state_message.channel = GAME_CHANNEL;
734     improvement_state_message.subject = "improvement state";
735 
736     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
737     improvement_state_message.int_payload["operation_maintenance_cost"] =
738         this->operation_maintenance_cost;
```

```
739
740     this->message_hub_ptr->sendMessage(improvement_state_message);
741
742     std::cout « "Improvement state message sent by " « this « std::endl;
743
744     return;
745 }   /* __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.

```
950 {
951     //  1. send improvement state message
952     this->__sendImprovementStateMessage();
953
954     //  2. update
955     this->__computeCapacityFactors();
956     this->update();
957
958     //  3. handle start/stop
959     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
960         this->is_running = true;
961     }
962
963     else if (this->is_running and (this->dispatch_MWh <= 0)) {
964         this->is_running = false;
965     }
966
967     //  4. handle equipment health and breakdowns
968     if (this->is_running) {
969         this->health--;
970
971         if (this->health <= 50) {
972             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
973
974             if ((double)rand() / RAND_MAX <= breakdown_prob) {
975                 this->health = 0;
976             }
977         }
978
979         if (this->health <= 0) {
980             this->__breakdown();
981         }
982     }
983
984     //  5. send tile state request (if selected)
985     if (this->is_selected) {
986         this->__sendTileStateRequest();
987     }
988
989     return;
990 }   /* 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.

```
1079 {
1080     //  1. if just built, call base method and return
1081     if (this->just_built) {
1082         TileImprovement :: draw();
1083 
1084         return;
1085     }
1086 
1087 
1088     //  2. handle upgrade effects
1089     if (this->just_upgraded) {
1090         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1091             this->tile_improvement_sprite_animated[i].setColor(
1092                 sf::Color(
1093                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1094                     255,
1095                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1096                     255
1097                 )
1098             );
1099 
1100             this->tile_improvement_sprite_animated[i].setScale(
1101                 sf::Vector2f(
1102                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1103                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1104                 )
1105             );
1106         }
1107 
1108         this->upgrade_frame++;
1109     }
1110 
1111     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1112         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1113             this->tile_improvement_sprite_animated[i].setColor(
1114                 sf::Color(255,255,255,255)
1115             );
1116 
1117             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1118         }
1119 
1120         this->just_upgraded = false;
1121         this->upgrade_frame = 0;
1122     }
1123 
1124 
1125     //  3. draw first element of animated sprite
1126     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1127 
1128 
1129     //  4. draw second element of animated sprite
1130     if (this->is_running) {
1131         this->tile_improvement_sprite_animated[0].setPosition(
1132             this->position_x,
1133             this->position_y + this->bobbing_y * cos(
1134                 (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1135             )
1136         );
1137 
1138         this->tile_improvement_sprite_animated[1].setPosition(
1139             this->position_x,
1140             this->position_y + 1.25 * this->bobbing_y * sin(
1141                 (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1142             )
1143         );
1144     }
1145 
1146     else {
1147         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1148             this->tile_improvement_sprite_animated[i].setPosition(
1149                 this->position_x,
1150                 this->position_y + this->bobbing_y * cos(
1151                     (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1152                 )
1153             );
```

```
1154         }
1155     }
1156
1157     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1158
1159
1160     //  5. draw storage upgrades
1161     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1162         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1163     }
1164
1165
1166     //  6. handle dispatch illustration
1167     if (this->dispatch_MWh > 0) {
1168         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1169         this->__drawDispatch();
1170     }
1171
1172
1173     //  7. draw production menu
1174     if (this->production_menu_open) {
1175         this->render_window_ptr->draw(this->production_menu_backing);
1176         this->render_window_ptr->draw(this->production_menu_backing_text);
1177
1178         this->__drawProductionMenu();
1179     }
1180
1181
1182     //  8. draw upgrade menu
1183     if (this->upgrade_menu_open) {
1184         this->render_window_ptr->draw(this->upgrade_menu_backing);
1185         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1186
1187         this->__drawUpgradeOptions();
1188     }
1189
1190
1191     //  9. handle broken effects
1192     if (this->is_broken) {
1193         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1194             this->tile_improvement_sprite_animated[i].setColor(
1195                 sf::Color(
1196                     255,
1197                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1198                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1199                     255
1200                 )
1201             );
1202         }
1203     }
1204
1205     this->frame++;
1206     return;
1207 }   /* 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.

```
862 {
863     //                  32 char x 17 line console "--------------------------------\n";
864     std::string options_substring                = "CAPACITY:      ";
865     options_substring                           += std::to_string(this->capacity_kW);
866     options_substring                           += " kW (level ";
867     options_substring                           += std::to_string(this->upgrade_level);
868     options_substring                           += ")\n";
```

```
869
870     options_substring                           += "PRODUCTION:    ";
871     options_substring                           += std::to_string(this->production_MWh);
872     options_substring                           += " MWh\n";
873
874     options_substring                           += "DISPATCHABLE:  ";
875     options_substring                           += std::to_string(this->dispatchable_MWh);
876     options_substring                           += " MWh\n";
877
878     options_substring                           += "HEALTH:      ";
879     options_substring                           += std::to_string(this->health);
880     options_substring                           += "/100";
881
882     if (this->health <= 0) {
883         options_substring                       += " ** BROKEN! **\n";
884     }
885
886     else {
887         options_substring                       += "\n";
888     }
889
890     options_substring                           += "                                \n";
891     options_substring                           += " **** WAVE ENERGY OPTIONS ****  \n";
892     options_substring                           += "                                \n";
893
894     if (this->is_broken) {
895         options_substring                       += "     [R]:  REPAIR (";
896         options_substring                       += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
897         options_substring                       += " K)\n";
898     }
899
900     else {
901         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
902     }
903
904     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
905     options_substring                           += "HOLD [P]:  SCRAP (";
906     options_substring                           += std::to_string(SCRAP_COST);
907     options_substring                           += " K)";
908
909     return options_substring;
910 }   /* getTileOptionsSubstring() */
```

4.14.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
1030 {
1031     TileImprovement :: processEvent();
1032
1033     if (this->event_ptr->type == sf::Event::KeyPressed) {
1034         this->__handleKeyPressEvents();
1035     }
1036
1037     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1038         this->__handleMouseButtonEvents();
1039     }
1040
1041     return;
1042 }   /* processEvent() */
```

4.14.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1057 {
1058     TileImprovement :: processMessage();
1059
1060     //...
1061
1062     return;
1063 }   /* 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.

```
927 {
928     TileImprovement :: setIsSelected(is_selected);
929
930     if (this->is_running and this->is_selected) {
931         this->assets_manager_ptr->getSound("ocean waves")->play();
932     }
933
934     return;
935 }   /* setIsSelected() */
```

4.14.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
1005 {
1006     this->__computeProduction();
1007     this->__computeProductionCosts();
1008     this->__computeDispatch();
1009
1010     if (this->is_selected) {
1011         this->__sendTileStateRequest();
1012     }
1013
1014     return;
1015 }   /* 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:

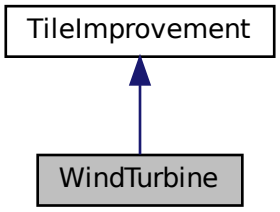
- header/WaveEnergyConverter.h
- source/WaveEnergyConverter.cpp

4.15 WindTurbine Class Reference

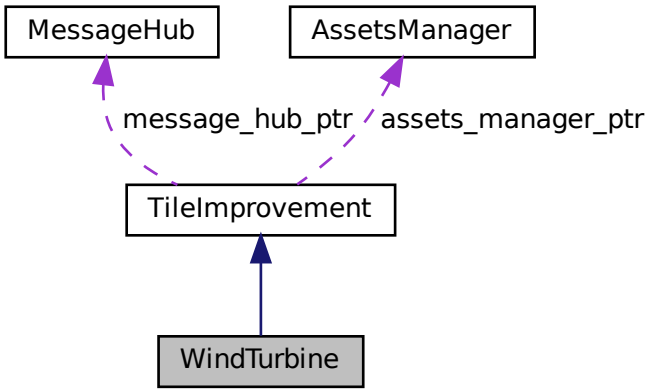
A settlement class (child class of TileImprovement).

`#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 (by definition in the closed interval [0, 1]).

- 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.

```
800   :
801 TileImprovement(
802     position_x,
803     position_y,
804     tile_resource,
805     event_ptr,
806     render_window_ptr,
807     assets_manager_ptr,
808     message_hub_ptr
809 )
810 {
811     //  1. set attributes
812
813     //  1.1. private
814     //...
815
816     //  1.2. public
817     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
818
819     this->is_running = false;
820
821     this->health = 100;
822
823     this->capacity_kW = 100;
824     this->upgrade_level = 1;
825
826     this->storage_kWh = 0;
827     this->storage_level = 0;
828
829     this->production_MWh = 0;
830     this->dispatch_MWh = 0;
831     this->dispatchable_MWh = 0;
832
833     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
834
835     this->rotor_drotation = 256 * SECONDS_PER_FRAME;
836
837     this->capacity_factor_vec.resize(30, 0);
838     this->production_vec_MWh.resize(30, 0);
839     this->dispatch_vec_MWh.resize(30, 0);
840
841     this->tile_improvement_string = "WIND TURBINE";
842
843     this->__setUpTileImprovementSpriteAnimated();
844     this->__computeCapacityFactors();
845     this->update();
846
847     std::cout « "WindTurbine constructed at " « this « std::endl;
848
849     return;
850 }   /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

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

Destructor for the WindTurbine class.

```
1205 {
1206     std::cout « "WindTurbine at " « this « " destroyed" « std::endl;
1207
1208     return;
1209 }   /* ~WindTurbine() */
```

4.15.3 Member Function Documentation

4.15.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
257 {
258     TileImprovement :: __breakdown();
259 
260     this->production_MWh = 0;
261     this->dispatch_MWh = 0;
262     this->dispatchable_MWh = 0;
263     this->operation_maintenance_cost = 0;
264 
265     return;
266 }   /* __breakdown() */
```

4.15.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors (by definition in the closed interval [0, 1]).

```
315 {
316     if (this->is_broken) {
317         return;
318     }
319 
320     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
321     std::default_random_engine generator(seed);
322 
323     double mean =
324         this->tile_resource_scalar * MEAN_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
325 
326     double stdev = STDEV_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
327 
328     if (this->tile_resource_scalar > 1) {
329         stdev /= this->tile_resource_scalar;
330     }
331 
332     double performance_factor = this->__getPerformanceFactor();
333 
334     std::normal_distribution<double> normal_dist(mean, stdev);
335 
336     double capacity_factor = 0;
337 
338     for (int i = 0; i < 30; i++) {
339         capacity_factor = performance_factor * normal_dist(generator);
340 
341         if (capacity_factor < 0) {
342             capacity_factor = 0;
343         }
344 
345         else if (capacity_factor > 1) {
346             capacity_factor = 1;
347         }
348 
349         this->capacity_factor_vec[i] = capacity_factor;
350     }
351 
352     return;
353 }   /* __computeCapacityFactors() */
```

4.15.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
401 {
402     if (this->is_broken) {
403         this->dispatchable_MWh = 0;
404         return;
405     }
406
407     double stored_energy_MWh = 0;
408     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
409
410     double demand_MWh = 0;
411     double production_MWh = 0;
412     double dispatchable_MWh = 0;
413     double difference_MWh = 0;
414
415     double room_MWh = 0;
416
417     for (int i = 0; i < 30; i++) {
418         demand_MWh = this->demand_vec_MWh[i];
419         production_MWh = this->production_vec_MWh[i];
420
421         if (production_MWh <= demand_MWh) {
422             this->dispatch_vec_MWh[i] = production_MWh;
423             dispatchable_MWh += this->dispatch_vec_MWh[i];
424
425             difference_MWh = demand_MWh - production_MWh;
426
427             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
428                 if (difference_MWh > stored_energy_MWh) {
429                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
430                     dispatchable_MWh += stored_energy_MWh;
431                     stored_energy_MWh = 0;
432                 }
433
434                 else {
435                     this->dispatch_vec_MWh[i] += difference_MWh;
436                     dispatchable_MWh += difference_MWh;
437                     stored_energy_MWh -= difference_MWh;
438                 }
439             }
440         }
441
442         else {
443             this->dispatch_vec_MWh[i] = demand_MWh;
444             dispatchable_MWh += this->dispatch_vec_MWh[i];
445
446             difference_MWh = production_MWh - demand_MWh;
447
448             if (
449                 (storage_capacity_MWh > 0) and
450                 (stored_energy_MWh < storage_capacity_MWh)
451             ) {
452                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
453
454                 if (difference_MWh > room_MWh) {
455                     stored_energy_MWh += room_MWh;
456                 }
457
458                 else {
459                     stored_energy_MWh += difference_MWh;
460                 }
461             }
462         }
463     }
464
465     this->dispatchable_MWh = round(dispatchable_MWh);
466
467     if (this->dispatch_MWh <= 0) {
468         this->dispatch_MWh = 0;
469     }
470
471     else if (this->dispatch_MWh != this->dispatchable_MWh) {
472         this->dispatch_MWh = this->dispatchable_MWh;
473     }
474
475     return;
476 }   /* __computeDispatch() */
```

4.15.3.4 __computeProduction()

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

Helper method to compute production values.

```
368 {
369     if (this->is_broken) {
370         this->production_MWh = 0;
371         return;
372     }
373
374     double production_MWh = 0;
375
376     for (int i = 0; i < 30; i++) {
377         this->production_vec_MWh[i] =
378             this->max_daily_production_MWh * this->capacity_factor_vec[i];
379
380         production_MWh += this->production_vec_MWh[i];
381     }
382
383     this->production_MWh = round(production_MWh);
384
385     return;
386 }   /* __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     if (this->is_running) {
231             double operation_maintenance_cost =
232         (this->production_MWh * WIND_OP_MAINT_COST_PER_MWH_PRODUCTION) / 1000;
233
234         this->operation_maintenance_cost = round(operation_maintenance_cost);
235     }
236
237     else {
238         this->operation_maintenance_cost = 0;
239     }
240
241     return;
242 }   /* __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.

```
617 {
618     //  1. draw power capacity upgrade sprite
619     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
620         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
621         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
622
623         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
624         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
625
626         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
627         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
628
629         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
630         this->tile_improvement_sprite_animated[i].setRotation(0);
631
632         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
633
634         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
635         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
636         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
637         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
638     }
639
640     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
641
642
643     //  2. draw power capacity upgrade text
644     //                  16 char line = "                \n"
645     std::string power_upgrade_string = "POWER CAPACITY  \n";
646     power_upgrade_string            += "                \n";
647
```

```
648     power_upgrade_string            += "CAPACITY:  ";
649     power_upgrade_string            += std::to_string(this->capacity_kW);
650     power_upgrade_string            += " kW\n";
651 
652     power_upgrade_string            += "LEVEL:     ";
653     power_upgrade_string            += std::to_string(this->upgrade_level);
654     power_upgrade_string            += "\n\n";
655 
656     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
657         power_upgrade_string        += "[W]: + 100 kW (";
658         power_upgrade_string        +=  std::to_string(WIND_TURBINE_BUILD_COST);
659         power_upgrade_string        += " K)\n";
660     }
661 
662     else {
663         power_upgrade_string        += " * MAX LEVEL *  \n";
664     }
665 
666     sf::Text power_upgrade_text = sf::Text(
667         power_upgrade_string,
668         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
669         16
670     );
671 
672     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
673     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
674     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
675 
676     this->render_window_ptr->draw(power_upgrade_text);
677 
678 
679     //  3. draw energy capacity (storage) upgrade sprite
680     this->render_window_ptr->draw(this->storage_upgrade_sprite);
681     this->render_window_ptr->draw(this->upgrade_plus_sprite);
682 
683 
684     //  4. draw energy capacity (storage) upgrade text
685     //                   16 char line = "               \n"
686     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
687     energy_upgrade_string            += "               \n";
688 
689     energy_upgrade_string            += "CAPACITY:  ";
690     energy_upgrade_string            += std::to_string(this->storage_level * 200);
691     energy_upgrade_string            += " kWh\n";
692 
693     energy_upgrade_string            += "LEVEL:     ";
694     energy_upgrade_string            += std::to_string(this->storage_level);
695     energy_upgrade_string            += "\n\n";
696 
697     if (this->storage_level < MAX_STORAGE_LEVELS) {
698         energy_upgrade_string        += "[D]: + 200 kWh (";
699         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
700         energy_upgrade_string        += " K)\n";
701     }
702 
703     else {
704         energy_upgrade_string += " * MAX LEVEL *  \n";
705     }
706 
707     sf::Text energy_upgrade_text = sf::Text(
708         energy_upgrade_string,
709         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
710         16
711     );
712 
713     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
714     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
715     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
716 
717     this->render_window_ptr->draw(energy_upgrade_text);
718 
719     return;
720 }   /* __drawUpgradeOptions() */
```

4.15.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
491 {
492     if (this->just_built) {
493         return;
494     }
495 
496     switch (this->event_ptr->key.code) {
497         case (sf::Keyboard::U): {
498             this->__openUpgradeMenu();
499 
500             break;
501         }
502 
503 
504         case (sf::Keyboard::W): {
505             if (this->production_menu_open) {
506                 this->dispatch_MWh++;
507 
508                 if (this->dispatch_MWh > this->dispatchable_MWh) {
509                     this->dispatch_MWh = 0;
510                 }
511 
512                 this->__computeProductionCosts();
513                 this->assets_manager_ptr->getSound("interface click")->play();
514             }
515 
516             else if (this->upgrade_menu_open) {
517                 this->__upgradePowerCapacity();
518             }
519 
520             break;
521         }
522 
523 
524         case (sf::Keyboard::S): {
525             if (this->production_menu_open) {
526                 this->dispatch_MWh--;
527 
528                 if (this->dispatch_MWh < 0) {
529                     this->dispatch_MWh = this->dispatchable_MWh;
530                 }
531 
532                 this->__computeProductionCosts();
533                 this->assets_manager_ptr->getSound("interface click")->play();
534             }
535 
536             break;
537         }
538 
539 
540         case (sf::Keyboard::D): {
541             if (this->upgrade_menu_open) {
542                 this->__upgradeStorageCapacity();
543                 this->__computeProduction();
544                 this->__computeDispatch();
545             }
546 
547             break;
548         }
549 
550 
551         default: {
552             // do nothing!
553 
554             break;
555         }
556     }
557 
558     return;
559 }   /* __handleKeyPressEvents() */
```

4.15.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
574 {
```

```
575     if (this->just_built) {
576         return;
577     }
578 
579     switch (this->event_ptr->mouseButton.button) {
580         case (sf::Mouse::Left): {
581             //...
582 
583             break;
584         }
585 
586 
587         case (sf::Mouse::Right): {
588             //...
589 
590             break;
591         }
592 
593 
594         default: {
595             // do nothing!
596 
597             break;
598         }
599     }
600 
601     return;
602 }   /* __handleMouseButtonEvents() */
```

4.15.3.10 __repair()

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

Helper method to repair the wind turbine.

Reimplemented from TileImprovement.

```
281 {
282     if (this->credits < WIND_TURBINE_BUILD_COST) {
283         std::cout « "Cannot repair wind turbine: insufficient credits (need "
284             « WIND_TURBINE_BUILD_COST « " K)" « std::endl;
285 
286         this->__sendInsufficientCreditsMessage();
287         return;
288     }
289 
290     TileImprovement :: __repair();
291 
292     this->just_upgraded = true;
293 
294     this->__sendCreditsSpentMessage(WIND_TURBINE_BUILD_COST);
295     this->__sendTileStateRequest();
296     this->__sendGameStateRequest();
297 
298     return;
299 }   /* __repair() */
```

4.15.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
735 {
736     Message improvement_state_message;
737 
738     improvement_state_message.channel = GAME_CHANNEL;
739     improvement_state_message.subject = "improvement state";
```

```
740
741     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
742     improvement_state_message.int_payload["operation_maintenance_cost"] =
743         this->operation_maintenance_cost;
744
745     this->message_hub_ptr->sendMessage(improvement_state_message);
746
747     std::cout « "Improvement state message sent by " « this « std::endl;
748
749     return;
750 }   /* __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.

```
955 {
956     //  1. send improvement state message
957     this->__sendImprovementStateMessage();
958
959     //  2. update
960     this->__computeCapacityFactors();
961     this->update();
962
963     //  3. handle start/stop
964     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
965         this->is_running = true;
966     }
967
968     else if (this->is_running and (this->dispatch_MWh <= 0)) {
969         this->is_running = false;
970     }
971
972     //  4. handle equipment health and breakdowns
973     if (this->is_running) {
974         this->health--;
975
976         if (this->health <= 50) {
977             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
978
979             if ((double)rand() / RAND_MAX <= breakdown_prob) {
980                 this->health = 0;
981             }
982         }
983
984         if (this->health <= 0) {
985             this->__breakdown();
986         }
987     }
988
989     //  5. send tile state request (if selected)
990     if (this->is_selected) {
991         this->__sendTileStateRequest();
992     }
993
994     return;
995 }   /* 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.

```
1086 {
1087     //  1. if just built, call base method and return
1088     if (this->just_built) {
1089         TileImprovement :: draw();
1090 
1091         return;
1092     }
1093 
1094     //  2. handle upgrade effects
1095     if (this->just_upgraded) {
1096         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1097             this->tile_improvement_sprite_animated[i].setColor(
1098                 sf::Color(
1099                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1100                     255,
1101                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1102                     255
1103                 )
1104             );
1105 
1106             this->tile_improvement_sprite_animated[i].setScale(
1107                 sf::Vector2f(
1108                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1109                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1110                 )
1111             );
1112         }
1113 
1114         this->upgrade_frame++;
1115     }
1116 
1117     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1118         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1119             this->tile_improvement_sprite_animated[i].setColor(
1120                 sf::Color(255,255,255,255)
1121             );
1122 
1123             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1124         }
1125 
1126         this->just_upgraded = false;
1127         this->upgrade_frame = 0;
1128     }
1129 
1130 
1131     //  3. draw first element of animated sprite
1132     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1133 
1134 
1135     //  4. draw second element of animated sprite
1136     if (this->is_running) {
1137         this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1138     }
1139 
1140     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1141 
1142 
1143     //  5. draw storage upgrades
1144     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1145         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1146     }
1147 
1148 
1149     //  6. handle dispatch illustration
1150     if (this->dispatch_MWh > 0) {
1151         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1152         this->__drawDispatch();
1153     }
1154 
1155 
1156     //  7. draw production menu
1157     if (this->production_menu_open) {
1158         this->render_window_ptr->draw(this->production_menu_backing);
1159         this->render_window_ptr->draw(this->production_menu_backing_text);
1160 
```

```
1161         this->__drawProductionMenu();
1162     }
1163
1164
1165     //  8. draw upgrade menu
1166     if (this->upgrade_menu_open) {
1167         this->render_window_ptr->draw(this->upgrade_menu_backing);
1168         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1169
1170         this->__drawUpgradeOptions();
1171     }
1172
1173
1174     //  9. handle broken effects
1175     if (this->is_broken) {
1176         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1177             this->tile_improvement_sprite_animated[i].setColor(
1178                 sf::Color(
1179                     255,
1180                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1181                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1182                     255
1183                 )
1184             );
1185         }
1186     }
1187
1188     this->frame++;
1189     return;
1190 }   /* 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.

```
867 {
868     //                  32 char x 17 line console "--------------------------------\n";
869     std::string options_substring                = "CAPACITY:      ";
870     options_substring                           += std::to_string(this->capacity_kW);
871     options_substring                           += " kW (level ";
872     options_substring                           += std::to_string(this->upgrade_level);
873     options_substring                           += ")\n";
874
875     options_substring                           += "PRODUCTION:    ";
876     options_substring                           += std::to_string(this->production_MWh);
877     options_substring                           += " MWh\n";
878
879     options_substring                           += "DISPATCHABLE:  ";
880     options_substring                           += std::to_string(this->dispatchable_MWh);
881     options_substring                           += " MWh\n";
882
883     options_substring                           += "HEALTH:      ";
884     options_substring                           += std::to_string(this->health);
885     options_substring                           += "/100";
886
887     if (this->health <= 0) {
888         options_substring                       += " ** BROKEN! **\n";
889     }
890
891     else {
892         options_substring                       += "\n";
893     }
894
895     options_substring                           += "                                \n";
896     options_substring                           += " **** WIND TURBINE OPTIONS **** \n";
897     options_substring                           += "                                \n";
```

```
898
899     if (this->is_broken) {
900         options_substring                       += "     [R]:  REPAIR (";
901         options_substring                       += std::to_string(WIND_TURBINE_BUILD_COST);
902         options_substring                       += " K)\n";
903     }
904
905     else {
906         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
907     }
908
909     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
910     options_substring                           += "HOLD [P]:  SCRAP (";
911     options_substring                           += std::to_string(SCRAP_COST);
912     options_substring                           += " K)";
913
914     return options_substring;
915 }   /* getTileOptionsSubstring() */
```

4.15.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
1037 {
1038     TileImprovement :: processEvent();
1039
1040     if (this->event_ptr->type == sf::Event::KeyPressed) {
1041         this->__handleKeyPressEvents();
1042     }
1043
1044     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1045         this->__handleMouseButtonEvents();
1046     }
1047
1048     return;
1049 }   /* processEvent() */
```

4.15.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1064 {
1065     TileImprovement :: processMessage();
1066
1067     //...
1068
1069     return;
1070 }   /* 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.

```
932 {
933     TileImprovement :: setIsSelected(is_selected);
934 
935     if (this->is_running and this->is_selected) {
936         this->assets_manager_ptr->getSound("wind turbine running")->play();
937     }
938 
939     return;
940 }   /* setIsSelected() */
```

4.15.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
1010 {
1011     std::cout « "WindTurbine :: update()" « std::endl;
1012 
1013     this->__computeProduction();
1014     this->__computeProductionCosts();
1015     this->__computeDispatch();
1016 
1017     if (this->is_selected) {
1018         this->__sendTileStateRequest();
1019     }
1020 
1021     return;
1022 }   /* 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

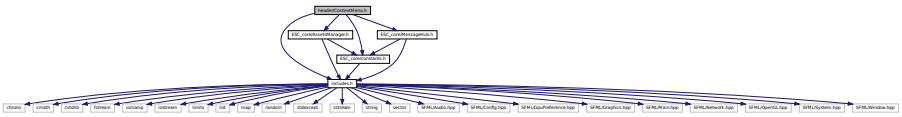
File Documentation

5.1 header/ContextMenu.h File Reference

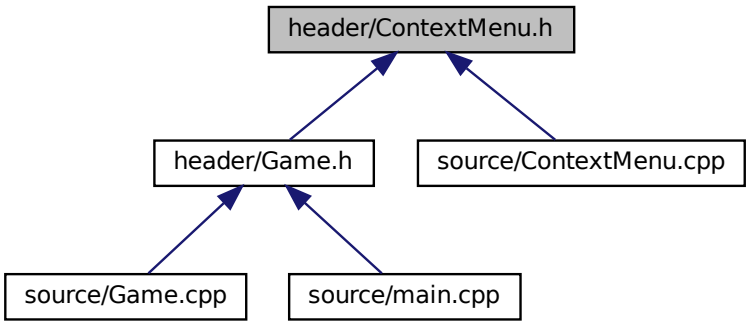
Header file for the ContextMenu class.

```
#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

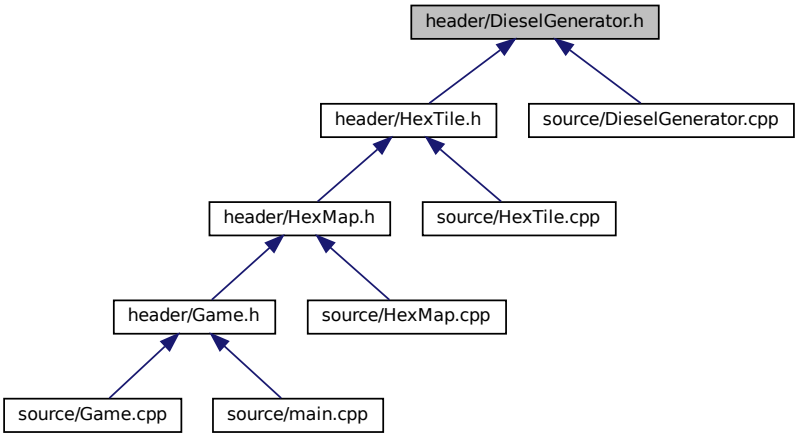
Header file for the DieselGenerator class.

```
#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.

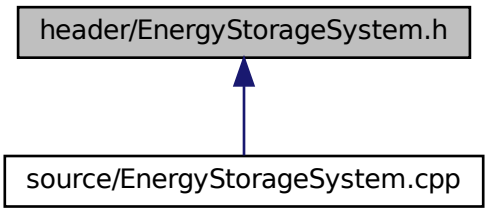
5.3 header/EnergyStorageSystem.h File Reference

Header file for the EnergyStorageSystem class. DEPRECATED / NOT USED.

```
#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).

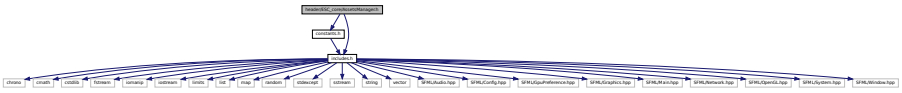
5.3.1 Detailed Description

Header file for the EnergyStorageSystem class. DEPRECATED / NOT USED.

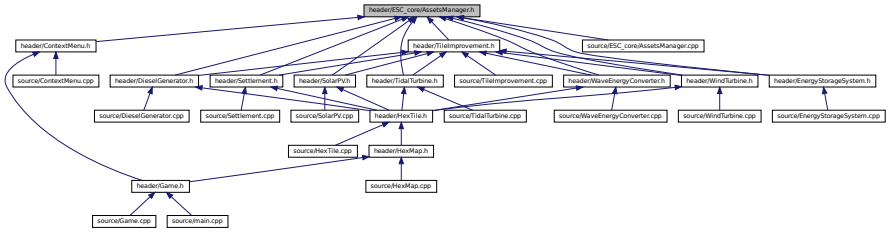
5.4 header/ESC_core/AssetsManager.h File Reference

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.

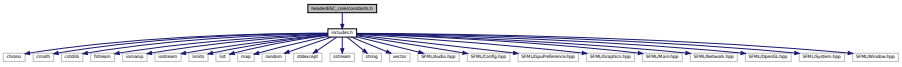
5.4.1 Detailed Description

Header file for the AssetsManager class.

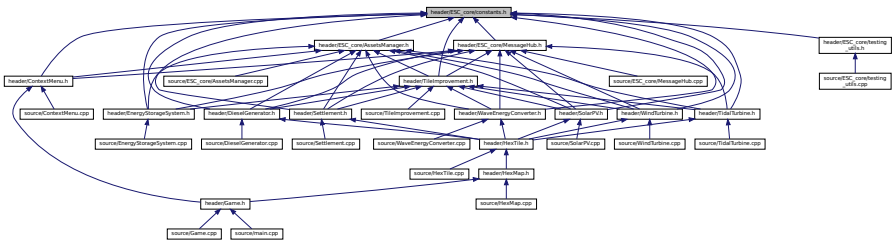
5.5 header/ESC_core/constants.h File Reference

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 std::string GAME_VERSION = "1.1"

  The current version of the game.
- const unsigned long long int SECONDS_PER_YEAR = 31537970

  The (average) number of seconds per year.
- const unsigned long long int SECONDS_PER_MONTH = 2628164

  The (average) number of seconds per month.
- 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 = 200

  The cost of building (or ugrading) a diesel generator in 200 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 double BREAKDOWN_PROBABILITY_INCREMENT = 0.01

  The amount by which equipment breakdown probability is incremented for each point of health below 50.
- const double PERFORMANCE_FACTOR_COEFFICIENT = 0.33771

  The coefficient of the performance factor curve (power expression).
- const double PERFORMANCE_FACTOR_EXPONENT = 0.23708

  The exponent of the performance factor curve (power expression).
- 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 = 1000

  The starting balance of credits.
- const double CREDITS_PER_MWH_SERVED = 1.15

  The number of credits (x1000) earned.
- const int EMISSIONS_LIFETIME_LIMIT_TONNES = 3200

  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 = 0.05

  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 = 0.275

  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.
- const std::vector< std::string > TUTORIAL_PAGES

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 BREAKDOWN_PROBABILITY_INCREMENT

```
const double BREAKDOWN_PROBABILITY_INCREMENT = 0.01
```

The amount by which equipment breakdown probability is incremented for each point of health below 50.

5.5.3.2 BUILD_SETTLEMENT_COST

```
const int BUILD_SETTLEMENT_COST = 250
```

The cost of building a settlement.

5.5.3.3 CLEAR_FOREST_COST

```
const int CLEAR_FOREST_COST = 160
```

The cost of clearing a forest tile.

5.5.3.4 CLEAR_MOUNTAINS_COST

```
const int CLEAR_MOUNTAINS_COST = 500
```

The cost of clearing a mountains tile.

5.5.3.5 CLEAR_PLAINS_COST

```
const int CLEAR_PLAINS_COST = 80
```

The cost of clearing a plains tile.

5.5.3.6 COST_PER_LITRE_DIESEL

```
const double COST_PER_LITRE_DIESEL = 1.75
```

The cost of a litre of diesel.

5.5.3.7 CREDITS_PER_MWH_SERVED

```
const double CREDITS_PER_MWH_SERVED = 1.15
```

The number of credits (x1000) earned.

5.5.3.8 DAILY_TIDAL_CAPACITY_FACTOR

```
const double DAILY_TIDAL_CAPACITY_FACTOR = 0.275
```

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).

5.5.3.9 DIESEL_GENERATOR_BUILD_COST

```
const int DIESEL_GENERATOR_BUILD_COST = 200
```

The cost of building (or ugrading) a diesel generator in 200 kW increments.

5.5.3.10 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).

5.5.3.11 EMISSIONS_LIFETIME_LIMIT_TONNES

```
const int EMISSIONS_LIFETIME_LIMIT_TONNES = 3200
```

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

5.5.3.12 ENERGY_STORAGE_SYSTEM_BUILD_COST

```
const int ENERGY_STORAGE_SYSTEM_BUILD_COST = 160
```

The cost of adding energy storage in 200 kWh increments.

5.5.3.13 FLOAT_TOLERANCE

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

Tolerance for floating point equality tests.

5.5.3.14 FRAMES_PER_SECOND

```
const int FRAMES_PER_SECOND = 60
```

Target frames per second.

5.5.3.15 GAME_CHANNEL

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

A message channel for game messages.

5.5.3.16 GAME_HEIGHT

```
const int GAME_HEIGHT = 800
```

Height of the game space.

5.5.3.17 GAME_STATE_CHANNEL

```
const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"
```

A message channel for game state messages.

5.5.3.18 GAME_VERSION

```
const std::string GAME_VERSION = "1.1"
```

The current version of the game.

5.5.3.19 GAME_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.20 HEX_MAP_CHANNEL

```
const std::string HEX_MAP_CHANNEL = "HEX MAP CHANNEL"
```

A message channel for hex map messages.

5.5.3.21 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.

5.5.3.22 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).

5.5.3.23 MAX_STORAGE_LEVELS

```
const int MAX_STORAGE_LEVELS = 5
```

The maximum storage level of any tile improvement.

5.5.3.24 MAX_UPGRADE_LEVELS

```
const int MAX_UPGRADE_LEVELS = 5
```

The maximum upgrade level of any tile improvement.

5.5.3.25 MAXIMUM_DAILY_DEMAND_PER_CAPITA

```
const double MAXIMUM_DAILY_DEMAND_PER_CAPITA = 0.05
```

The maximum daily demand [MWh] (at any point in the year) per capita.

5.5.3.26 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.

5.5.3.27 MEAN_DAILY_SOLAR_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.040, 0.083, 0.160,
    0.250, 0.312, 0.319,
    0.300, 0.253, 0.190,
    0.111, 0.055, 0.027
}
```

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.

5.5.3.28 MEAN_DAILY_WAVE_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.483, 0.451, 0.402,
    0.304, 0.238, 0.190,
    0.182, 0.191, 0.243,
    0.276, 0.431, 0.390
}
```

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.

5.5.3.29 MEAN_DAILY_WIND_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.360, 0.361, 0.382,
    0.383, 0.352, 0.327,
    0.269, 0.308, 0.357,
    0.376, 0.372, 0.353
}
```

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.

5.5.3.30 MEAN_POPULATION_GROWTH_RATE

```
const double MEAN_POPULATION_GROWTH_RATE = 0.020
```

The mean monthly population growth rate.

5.5.3.31 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

5.5.3.32 PERFORMANCE_FACTOR_COEFFICIENT

```
const double PERFORMANCE_FACTOR_COEFFICIENT = 0.33771
```

The coefficient of the performance factor curve (power expression).

5.5.3.33 PERFORMANCE_FACTOR_EXPONENT

```
const double PERFORMANCE_FACTOR_EXPONENT = 0.23708
```

The exponent of the performance factor curve (power expression).

5.5.3.34 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.35 SCRAP_COST

```
const int SCRAP_COST = 50
```

The cost of scrapping a tile improvement (other than settlement).

5.5.3.36 SECONDS_PER_FRAME

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

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

5.5.3.37 SECONDS_PER_MONTH

```
const unsigned long long int SECONDS_PER_MONTH = 2628164
```

The (average) number of seconds per month.

5.5.3.38 SECONDS_PER_YEAR

```
const unsigned long long int SECONDS_PER_YEAR = 31537970
```

The (average) number of seconds per year.

5.5.3.39 SETTLEMENT_CHANNEL

```
const std::string SETTLEMENT_CHANNEL = "SETTLEMENT CHANNEL"
```

A message channel for the settlement.

5.5.3.40 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).

5.5.3.41 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.

5.5.3.42 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.285714
```

The additional cost of building on water.

5.5.3.43 STARTING_CREDITS

```
const int STARTING_CREDITS = 1000
```

The starting balance of credits.

5.5.3.44 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.45 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.

5.5.3.46 STDEV_DAILY_SOLAR_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.018, 0.033, 0.059,
    0.067, 0.099, 0.099,
    0.104, 0.089, 0.066,
    0.036, 0.025, 0.012
}
```

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.

5.5.3.47 STDEV_DAILY_WAVE_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.095, 0.088, 0.106,
    0.094, 0.103, 0.069,
    0.055, 0.038, 0.094,
    0.111, 0.120, 0.201
}
```

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.

5.5.3.48 STDEV_DAILY_WIND_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.089, 0.086, 0.120,
    0.094, 0.099, 0.123,
    0.109, 0.132, 0.120,
    0.103, 0.085, 0.103
}
```

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.

5.5.3.49 STDEV_POPULATION_GROWTH_RATE

```
const double STDEV_POPULATION_GROWTH_RATE = 0.005
```

The standard deviation in monthly population growth rate.

5.5.3.50 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).

5.5.3.51 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 550
```

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

5.5.3.52 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).

5.5.3.53 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.54 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.55 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).

5.5.3.56 TUTORIAL_PAGES

```
const std::vector<std::string> TUTORIAL_PAGES
```

5.5.3.57 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.

5.5.3.58 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).

5.5.3.59 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).

5.5.3.60 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 450
```

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

5.5.3.61 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.

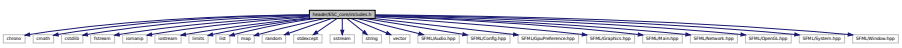
Ref: van Heesch. [2023]

5.7 header/ESC_core/includes.h File Reference

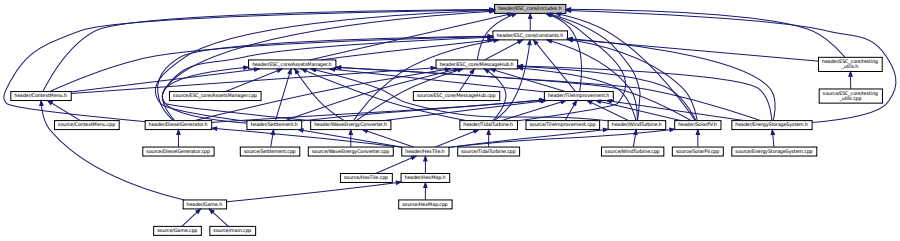
Header file for various includes.

```
#include <chrono>
#include <cmath>
#include <cstdlib>
#include <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.

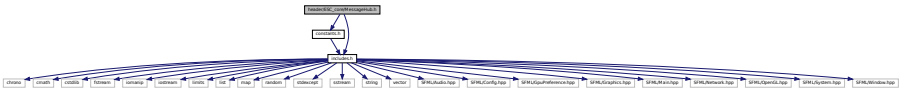
Ref: Gomila [2023]

5.8 header/ESC_core/MessageHub.h File Reference

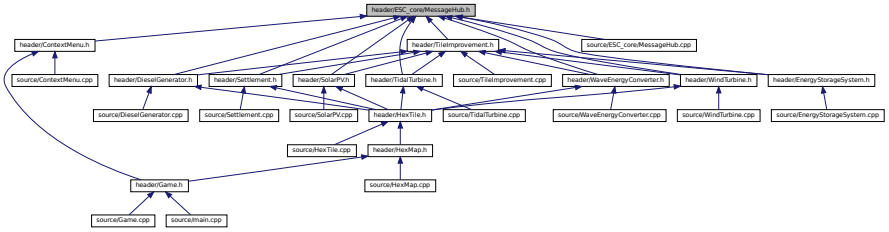
Header file for the MessageHub class.

```
#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

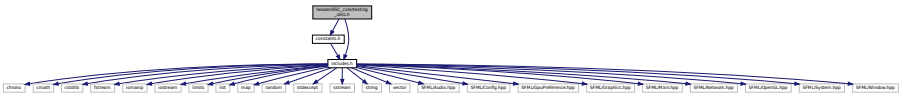
Header file for the MessageHub class.

5.9 header/ESC_core/testing_utils.h File Reference

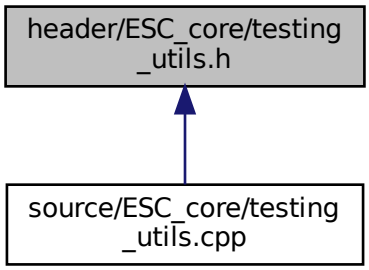
Header file for various testing utilities.

```
#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__").

```
434 {
435     std::string error_str = "\n ERROR  failed to throw expected error prior to line ";
436     error_str += std::to_string(line);
437     error_str += " of ";
438     error_str += file;
439 
440     #ifdef _WIN32
441         std::cout << error_str << std::endl;
442     #endif
443 
444     throw std::runtime_error(error_str);
445     return;
446 }   /* 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.

```
86 {
87     std::cout « "\x1B[33m" « input_str « "\033[0m";
88     return;
89 }   /* 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.

```
66 {
67     std::cout « "\x1B[32m" « input_str « "\033[0m";
68     return;
69 }   /* 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.

```
106 {
107     std::cout « "\x1B[31m" « input_str « "\033[0m";
108     return;
109 }   /* 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__").

```
140 {
141     if (fabs(x - y) <= FLOAT_TOLERANCE) {
142         return;
143     }
144 
145     std::string error_str = "ERROR: testFloatEquals():\t in ";
146     error_str += file;
147     error_str += "\tline ";
148     error_str += std::to_string(line);
149     error_str += ":\t\n";
150     error_str += std::to_string(x);
151     error_str += " and ";
152     error_str += std::to_string(y);
153     error_str += " are not equal to within +/- ";
154     error_str += std::to_string(FLOAT_TOLERANCE);
155     error_str += "\n";
156 
157     #ifdef _WIN32
158         std::cout << error_str << std::endl;
159     #endif
160 
161     throw std::runtime_error(error_str);
162     return;
163 }   /* 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__").

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

```
211
212     throw std::runtime_error(error_str);
213     return;
214 }   /* 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__").

```
244 {
245     if (x >= y) {
246         return;
247     }
248
249     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
250     error_str += file;
251     error_str += "\tline ";
252     error_str += std::to_string(line);
253     error_str += ":\t\n";
254     error_str += std::to_string(x);
255     error_str += " is not greater than or equal to ";
256     error_str += std::to_string(y);
257     error_str += "\n";
258
259     #ifdef _WIN32
260         std::cout « error_str « std::endl;
261     #endif
262
263     throw std::runtime_error(error_str);
264     return;
265 }   /* 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.

Parameters

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

```
295 {
296     if (x < y) {
297         return;
298     }
299
300     std::string error_str = "ERROR: testLessThan():\t in ";
301     error_str += file;
302     error_str += "\tline ";
303     error_str += std::to_string(line);
304     error_str += ":\t\n";
305     error_str += std::to_string(x);
306     error_str += " is not less than ";
307     error_str += std::to_string(y);
308     error_str += "\n";
309
310     #ifdef _WIN32
311         std::cout « error_str « std::endl;
312     #endif
313
314     throw std::runtime_error(error_str);
315     return;
316 }   /* 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__").

```
346 {
347     if (x <= y) {
348         return;
349     }
350
351     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
352     error_str += file;
353     error_str += "\tline ";
354     error_str += std::to_string(line);
355     error_str += ":\t\n";
356     error_str += std::to_string(x);
357     error_str += " is not less than or equal to ";
358     error_str += std::to_string(y);
359     error_str += "\n";
360
361     #ifdef _WIN32
362         std::cout « error_str « std::endl;
363     #endif
364
365     throw std::runtime_error(error_str);
366     return;
```

```
367 }   /* testLessThanOrEqualTo() */
```

5.9.2.10 testTruth()

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

Tests if the given statement is true.

Parameters

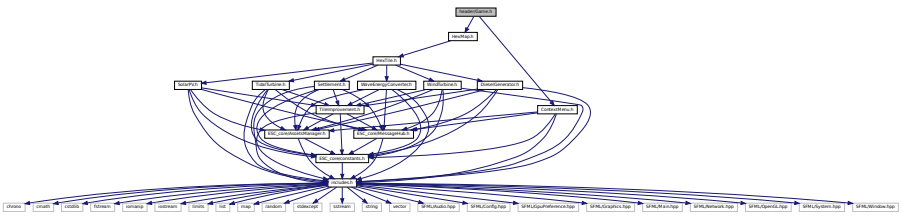
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
394 {
395     if (statement) {
396         return;
397     }
398
399     std::string error_str = "ERROR: testTruth():\t in ";
400     error_str += file;
401     error_str += "\tline ";
402     error_str += std::to_string(line);
403     error_str += ":\t\n";
404     error_str += "Given statement is not true";
405
406     #ifdef _WIN32
407         std::cout « error_str « std::endl;
408     #endif
409
410     throw std::runtime_error(error_str);
411     return;
412 }   /* testTruth() */
```

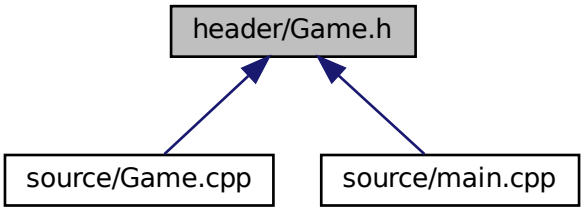
5.10 header/Game.h File Reference

```
#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

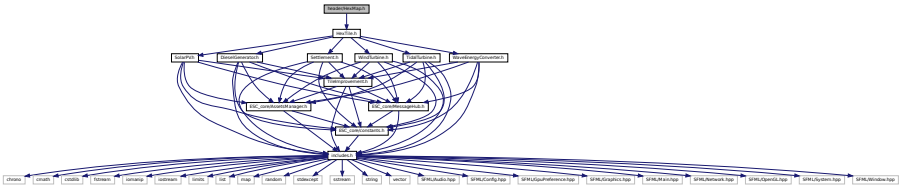
BUILD_SETTLEMENT	The settlement building phase.
SYSTEM_MANAGEMENT	The system management phase (main phase of play).
LOSS_EMISSIONS	A loss due to excessive emissions.
LOSS_DEMAND	A loss due to failing to meet the demand.
LOSS_CREDITS	A loss due to running out of credits.
VICTORY	A victory (12 consecutive 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 */
```

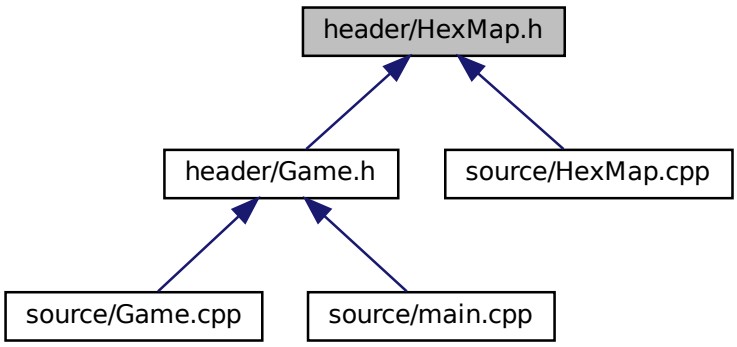
5.11 header/HexMap.h File Reference

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

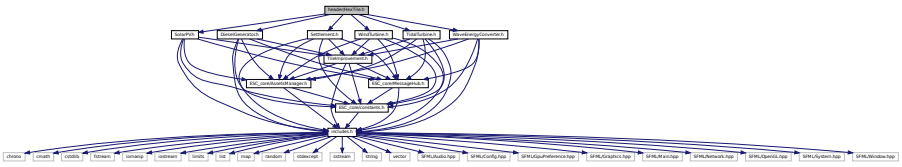
Header file for the HexMap class.

5.12 header/HexTile.h File Reference

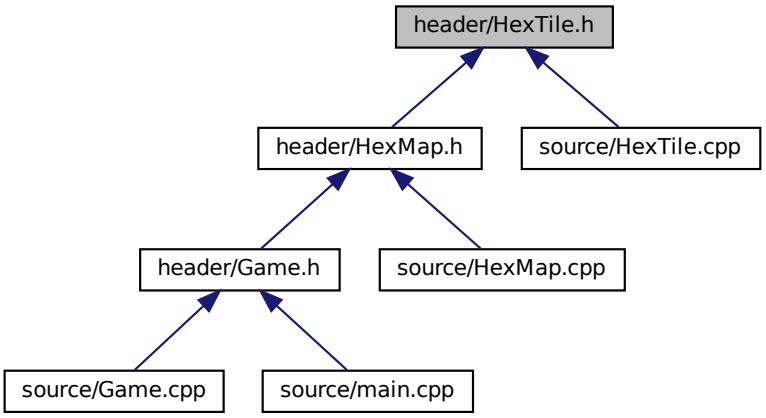
Header file for the Game class.

```
#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.

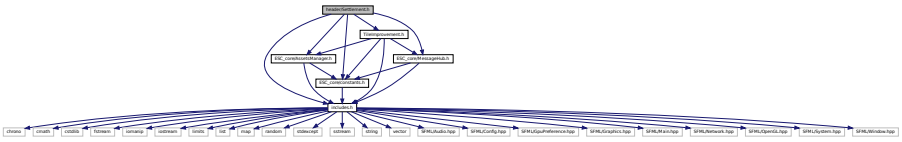
```
71              {
72     NONE_TYPE,
73     FOREST,
74     LAKE,
75     MOUNTAINS,
76     OCEAN,
77     PLAINS,
78     N_TILE_TYPES
79 }; /* TileType */
```

5.13 header/Settlement.h File Reference

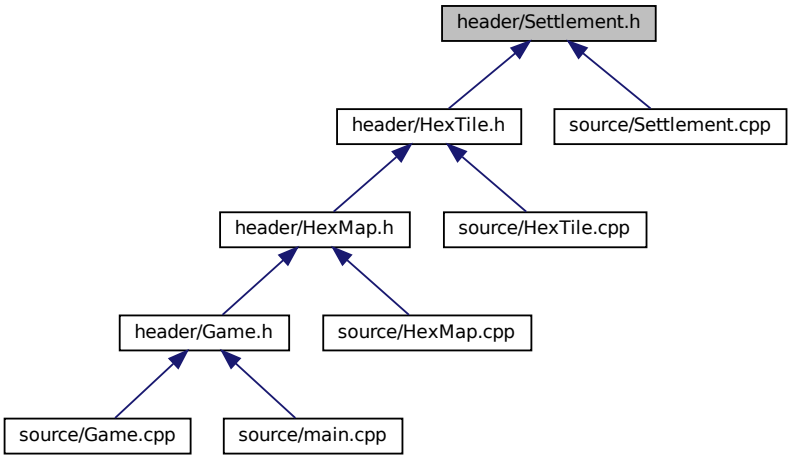
Header file for the Settlement class.

```
#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.

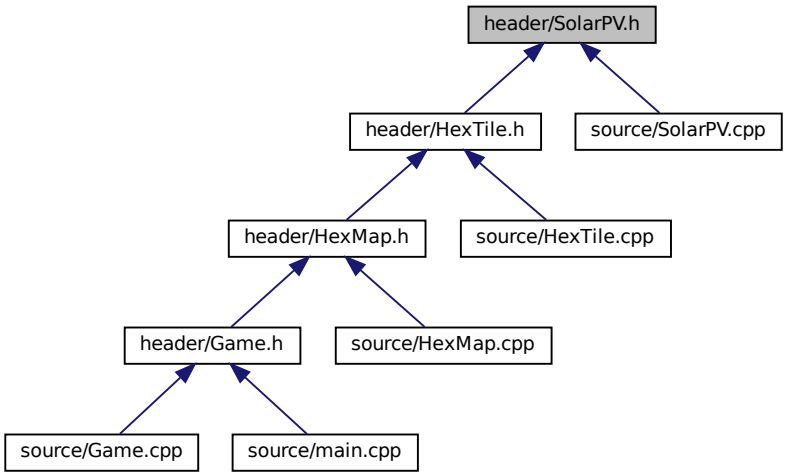
5.14 header/SolarPV.h File Reference

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

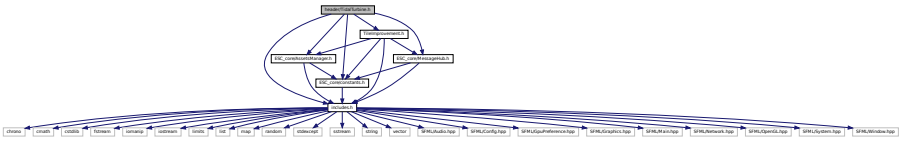
Header file for the SolarPV class.

5.15 header/TidalTurbine.h File Reference

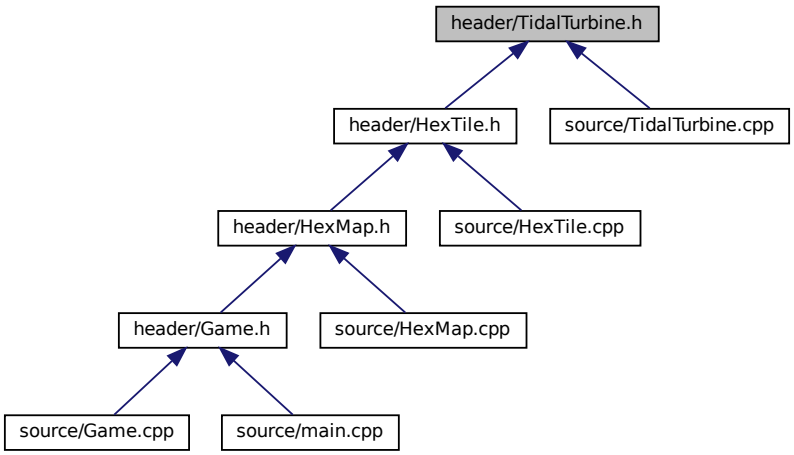
Header file for the TidalTurbine class.

```
#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

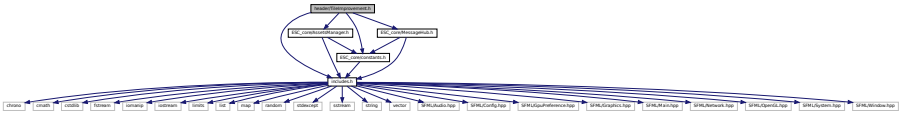
Header file for the TidalTurbine class.

5.16 header/TileImprovement.h File Reference

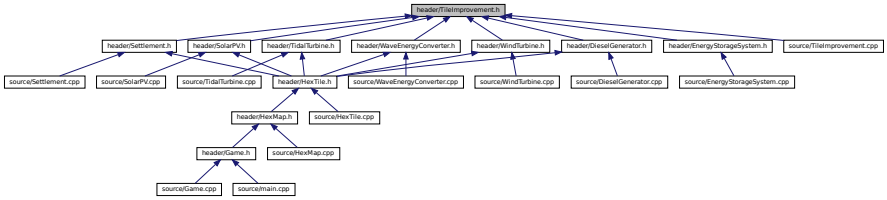
Header file for the TileImprovement class.

```
#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.

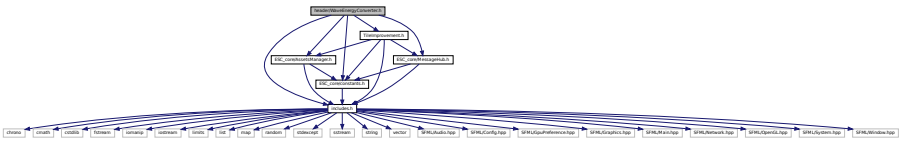
```
68                          {
69     SETTLEMENT,
70     DIESEL_GENERATOR,
71     SOLAR_PV,
72     WIND_TURBINE,
73     TIDAL_TURBINE,
74     WAVE_ENERGY_CONVERTER,
75     N_TILE_IMPROVEMENT_TYPES
76 };  /* TileImprovementType */
```

5.17 header/WaveEnergyConverter.h File Reference

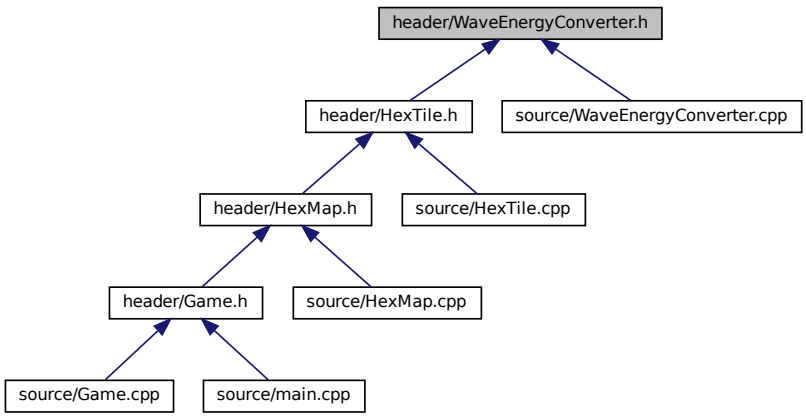
Header file for the WaveEnergyConverter class.

```
#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

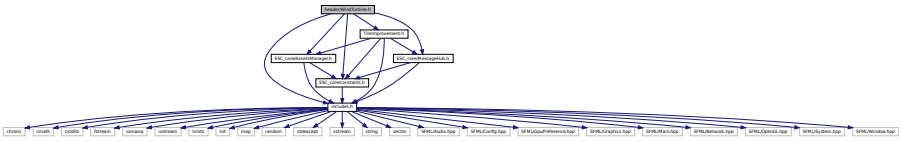
Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

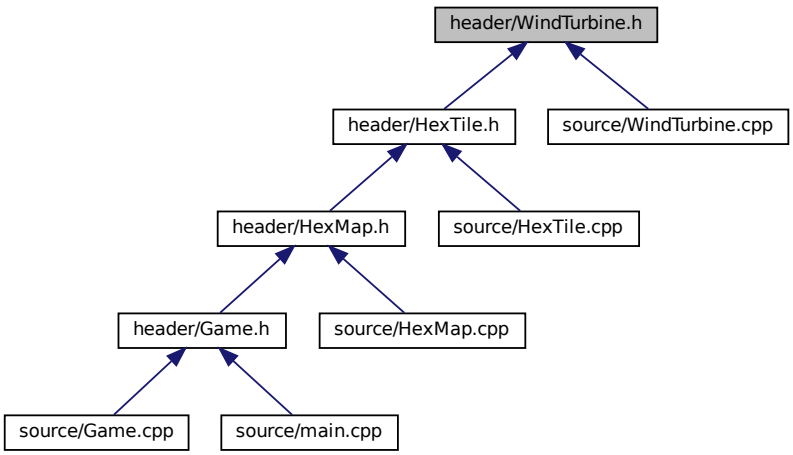
Header file for the WindTurbine class.

```
#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).

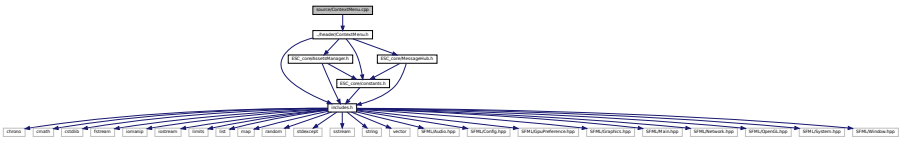
5.18.1 Detailed Description

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

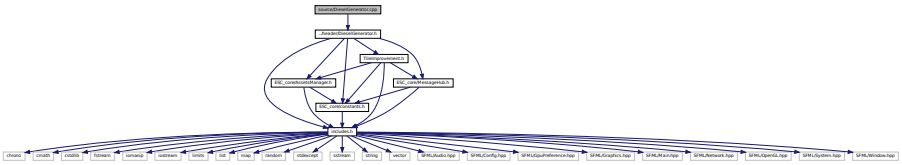
Implementation file for the ContextMenu class.

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

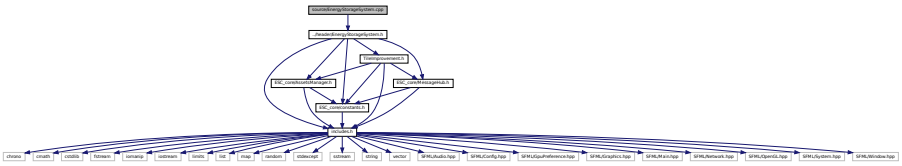
Implementation file for the DieselGenerator class.

A base class for the tile improvement hierarchy.

5.21 source/EnergyStorageSystem.cpp File Reference

Implementation file for the EnergyStorageSystem class. DEPRECATED / NOT USED.

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



5.21.1 Detailed Description

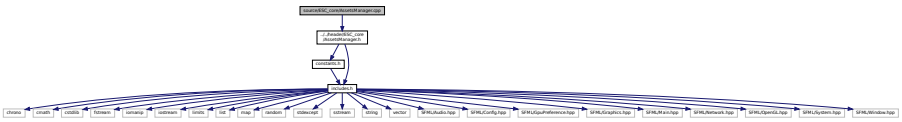
Implementation file for the EnergyStorageSystem class. DEPRECATED / NOT USED.

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

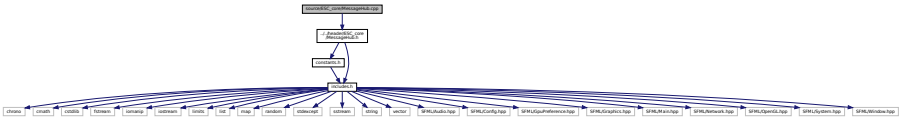
Implementation file for the AssetsManager class.

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.

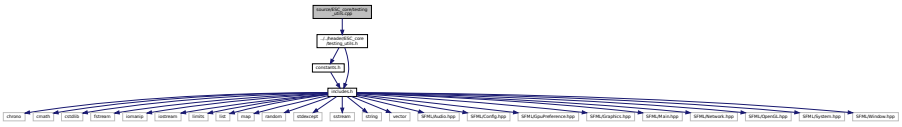
A class which acts as a central hub for inter-object message traffic.

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__").

```
434 {
435     std::string error_str = "\n ERROR  failed to throw expected error prior to line ";
436     error_str += std::to_string(line);
437     error_str += " of ";
438     error_str += file;
439 
440     #ifdef _WIN32
441         std::cout « error_str « std::endl;
442     #endif
443 
444     throw std::runtime_error(error_str);
445     return;
446 }   /* 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.

```
86 {
87     std::cout « "\x1B[33m" « input_str « "\033[0m";
88     return;
89 }   /* 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.

```
66 {
67     std::cout « "\x1B[32m" « input_str « "\033[0m";
68     return;
69 }   /* 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.

```
106 {
107     std::cout « "\x1B[31m" « input_str « "\033[0m";
108     return;
109 }   /* 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__").

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

```
154     error_str += std::to_string(FLOAT_TOLERANCE);
155     error_str += "\n";
156 
157     #ifdef _WIN32
158         std::cout « error_str « std::endl;
159     #endif
160 
161     throw std::runtime_error(error_str);
162     return;
163 }   /* 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__").

```
193 {
194     if (x > y) {
195         return;
196     }
197 
198     std::string error_str = "ERROR: testGreaterThan():\t in ";
199     error_str += file;
200     error_str += "\tline ";
201     error_str += std::to_string(line);
202     error_str += ":\t\n";
203     error_str += std::to_string(x);
204     error_str += " is not greater than ";
205     error_str += std::to_string(y);
206     error_str += "\n";
207 
208     #ifdef _WIN32
209         std::cout « error_str « std::endl;
210     #endif
211 
212     throw std::runtime_error(error_str);
213     return;
214 }   /* 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__").

```
244 {
245     if (x >= y) {
246         return;
247     }
248
249     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
250     error_str += file;
251     error_str += "\tline ";
252     error_str += std::to_string(line);
253     error_str += ":\t\n";
254     error_str += std::to_string(x);
255     error_str += " is not greater than or equal to ";
256     error_str += std::to_string(y);
257     error_str += "\n";
258
259     #ifdef _WIN32
260         std::cout « error_str « std::endl;
261     #endif
262
263     throw std::runtime_error(error_str);
264     return;
265 }   /* 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__").

```
295 {
296     if (x < y) {
297         return;
298     }
299
300     std::string error_str = "ERROR: testLessThan():\t in ";
301     error_str += file;
302     error_str += "\tline ";
303     error_str += std::to_string(line);
304     error_str += ":\t\n";
305     error_str += std::to_string(x);
306     error_str += " is not less than ";
307     error_str += std::to_string(y);
308     error_str += "\n";
309
310     #ifdef _WIN32
311         std::cout « error_str « std::endl;
312     #endif
313
314     throw std::runtime_error(error_str);
```

```
315     return;
316 }   /* 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__").

```
346 {
347     if (x <= y) {
348         return;
349     }
350 
351     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
352     error_str += file;
353     error_str += "\tline ";
354     error_str += std::to_string(line);
355     error_str += ":\t\n";
356     error_str += std::to_string(x);
357     error_str += " is not less than or equal to ";
358     error_str += std::to_string(y);
359     error_str += "\n";
360 
361     #ifdef _WIN32
362         std::cout « error_str « std::endl;
363     #endif
364 
365     throw std::runtime_error(error_str);
366     return;
367 }   /* testLessThanOrEqualTo() */
```

5.24.2.10 testTruth()

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

Tests if the given statement is true.

Parameters

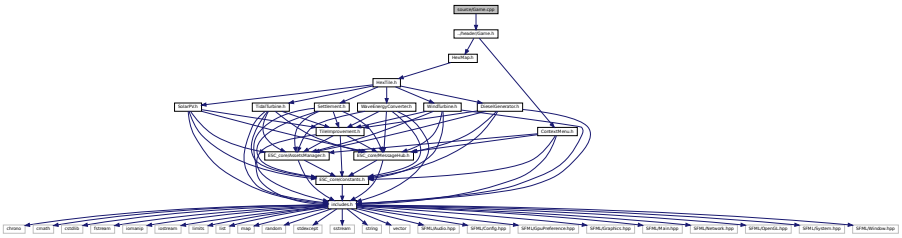
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
394 {
395     if (statement) {
396         return;
397     }
398
399     std::string error_str = "ERROR: testTruth():\t in ";
400     error_str += file;
401     error_str += "\tline ";
402     error_str += std::to_string(line);
403     error_str += ":\t\n";
404     error_str += "Given statement is not true";
405
406     #ifdef _WIN32
407         std::cout « error_str « std::endl;
408     #endif
409
410     throw std::runtime_error(error_str);
411     return;
412 }   /* 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

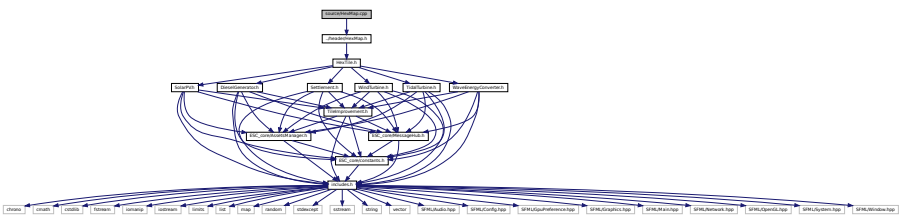
Implementation file for the Game class.

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

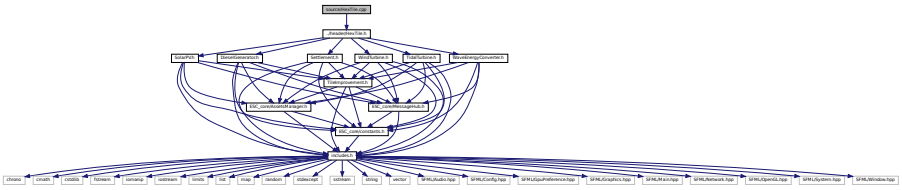
Implementation file for the HexMap class.

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

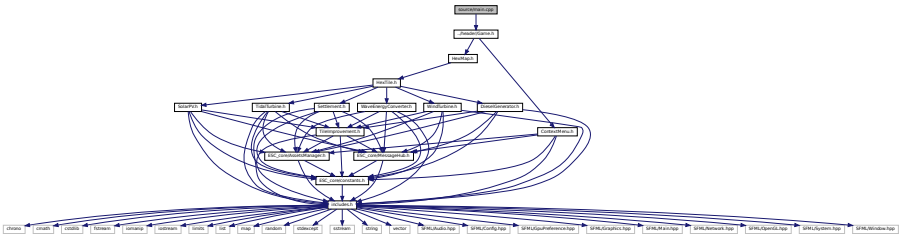
Implementation file for the HexTile class.

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.
- void playBrandAnimation (sf::RenderWindow *render_window_ptr)
- void showTitleScreen (sf::RenderWindow *render_window_ptr, AssetsManager *assets_manager_ptr)

  Helper function.
- 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.

```
345 {
346     //  1. get desktop resolution
347     sf::VideoMode video_mode = sf::VideoMode::getDesktopMode();
348     int desktop_width = video_mode.width;
349     int desktop_height = video_mode.height;
350 
351     //  2. adjust render window dimensions as necessary (maintain 3:2 aspect ratio)
352     int window_width = GAME_WIDTH;
353     int window_height = GAME_HEIGHT;
354 
355     if (
356         (window_width > desktop_width) or
357         (window_height > desktop_height)
358     ) {
359         int width_diff = window_width - desktop_width;
360         int height_diff = window_height - desktop_height;
361 
362         if (width_diff > height_diff) {
363             window_width = desktop_width;
364             window_height = (2.0 / 3.0) * desktop_width;
365         }
366 
367         else {
368             window_height = desktop_height;
369             window_width = (3.0 / 2.0) * desktop_height;
370         }
371     }
372 
373     //  3. construct render window
374     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
375         sf::VideoMode(window_width, window_height),
376         "Road To Zero"
377     );
```

```
378
379     //  4. reset render window view as necessary
380     if (
381         (window_width != GAME_WIDTH) or
382         (window_height != GAME_HEIGHT)
383     ) {
384         sf::View view;
385         view.reset(sf::FloatRect(0, 0, GAME_WIDTH, GAME_HEIGHT));
386         render_window_ptr->setView(view);
387     }
388
389     //  5. change render window icon
390     sf::Image icon;
391     icon.loadFromFile("assets/images/RoadToZero_favicon_CC-BY.png");
392
393     render_window_ptr->setIcon(
394         icon.getSize().x,
395         icon.getSize().y,
396         icon.getPixelsPtr()
397     );
398
399     return render_window_ptr;
400 }   /* 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     assets_manager_ptr->loadSound(
306         "assets/audio/samples/mixkit-poker-card-flick-2002_MixkitFree.ogg",
307         "card flick"
308     );
309
310
311     //  4. load tracks
312     assets_manager_ptr->loadTrack(
313         "assets/audio/tracks/TreeStarMoon_Dobranoc_CC0.ogg",
314         "Tree Star Moon - Dobranoc"
315     );
316
317     assets_manager_ptr->loadTrack(
318         "assets/audio/tracks/TreeStarMoon_Lighthouse_CC0.ogg",
319         "Tree Star Moon - Lighthouse"
320     );
321
322     assets_manager_ptr->loadTrack(
323         "assets/audio/tracks/TreeStarMoon_SkyFarm_CC0.ogg",
324         "Tree Star Moon - Sky Farm"
325     );
326
327     return;
328 }   /* loadAssets() */
```

5.28.2.3 main()

```
int main (
              int argc,
              char ** argv )
1002 {
1003     //  1. load assets
1004     AssetsManager assets_manager;
1005     loadAssets(&assets_manager);
1006
1007     //  2. construct render window
1008     sf::RenderWindow* render_window_ptr = constructRenderWindow();
1009
1010     //  3. show brand animation and splash screen
1011     playBrandAnimation(render_window_ptr);
1012
1013     //  4. show title screen
1014     if (render_window_ptr->isOpen()) {
1015         showTitleScreen(render_window_ptr, &assets_manager);
1016     }
1017
1018     //  5. start game loop
1019     bool quit_game = false;
1020     bool transition_from_title = true;
1021
1022     if (render_window_ptr->isOpen()) {
1023         assets_manager.playTrack();
1024     }
```

```
1025
1026     else {
1027         quit_game = true;
1028     }
1029
1030     while (not quit_game) {
1031         Game game(render_window_ptr, &assets_manager, transition_from_title);
1032         quit_game = game.run();
1033
1034         if (transition_from_title and (not quit_game)) {
1035             transition_from_title = false;
1036         }
1037     }
1038
1039     //  4. clean up
1040     render_window_ptr->close();
1041     delete render_window_ptr;
1042
1043     return 0;
1044 }   /* main() */
```

5.28.2.4 playBrandAnimation()

```
void playBrandAnimation (
             sf::RenderWindow * render_window_ptr )
418 {
419     //  1. load assets
420     AssetsManager brand_assets_manager;
421
422     brand_assets_manager.loadFont(
423         "assets/ESC_brand/OpenSans-Bold.ttf",
424         "OpenSansBold"
425     );
426
427     brand_assets_manager.loadTexture(
428         "assets/ESC_brand/ESC_key_109x90.png",
429         "[ESC] large"
430     );
431
432     brand_assets_manager.loadTexture(
433         "assets/ESC_brand/ESC_key_98x81.png",
434         "[ESC] small"
435     );
436
437     brand_assets_manager.loadTexture(
438         "assets/ESC_brand/SFML_256x128.png",
439         "SFML"
440     );
441
442     brand_assets_manager.loadSound(
443         "assets/ESC_brand/mixkit-single-key-type-2533_MixkitFree.ogg",
444         "key press"
445     );
446
447
448     //  2. set up and position assets
449     std::string brand_string = "INTERACTIVE";
450
451     sf::Text brand_text(
452         brand_string,
453         *(brand_assets_manager.getFont("OpenSansBold")),
454         64
455     );
456
457     brand_text.setOrigin(
458         brand_text.getLocalBounds().width / 2,
459         brand_text.getLocalBounds().height / 2
460     );
461
462     brand_text.setPosition(GAME_WIDTH / 2, GAME_HEIGHT / 2);
463
464     double key_position_x =
465         (GAME_WIDTH / 2) - (brand_text.getLocalBounds().width / 2) - 64;
466
467     double key_position_y =
468         (GAME_HEIGHT / 2) - (brand_text.getLocalBounds().height / 2) - 32;
469
470     sf::Sprite ESC_large(
471         *(brand_assets_manager.getTexture("[ESC] large"))
```

```
472     );
473
474     ESC_large.setOrigin(
475         ESC_large.getLocalBounds().width / 2,
476         ESC_large.getLocalBounds().height / 2
477     );
478
479     ESC_large.setPosition(key_position_x, key_position_y);
480
481     ESC_large.setColor(sf::Color(255, 255, 255, 0));
482
483     sf::Sprite ESC_small(
484         *(brand_assets_manager.getTexture("[ESC] small"))
485     );
486
487     ESC_small.setOrigin(
488         ESC_small.getLocalBounds().width / 2,
489         ESC_small.getLocalBounds().height / 2
490     );
491
492     ESC_small.setPosition(key_position_x, key_position_y);
493
494     ESC_small.setColor(sf::Color(255, 255, 255, 255));
495
496     sf::Sprite SFML(
497         *(brand_assets_manager.getTexture("SFML"))
498     );
499
500     SFML.setOrigin(
501         SFML.getLocalBounds().width / 2,
502         SFML.getLocalBounds().height / 2
503     );
504
505     SFML.setPosition(GAME_WIDTH / 2, GAME_HEIGHT / 2);
506
507     SFML.setColor(sf::Color(255, 255, 255, 0));
508
509
510     //  3. draw loop
511     bool sound_played = false;
512
513     int brand_frame = 0;
514     int click_frame = 0;
515     int brand_state = 0;
516
517     size_t substring_idx = 0;
518
519     double alpha = 0;
520     double dalpha = FRAMES_PER_SECOND / 18;
521     double time_since_start_s = 0;
522
523     sf::Clock brand_clock;
524     sf::Event brand_event;
525
526     while (brand_state < 6) {
527         time_since_start_s = brand_clock.getElapsedTime().asSeconds();
528
529         if (time_since_start_s >= (brand_frame + 1) * SECONDS_PER_FRAME) {
530             render_window_ptr->clear();
531
532             while (render_window_ptr->pollEvent(brand_event)) {
533                 if (brand_event.type == sf::Event::Closed) {
534                     render_window_ptr->close();
535                     return;
536                 }
537             }
538
539             //  3.1. brand state switch
540             switch (brand_state) {
541                 case (0): {
542                     // fade in key
543                     render_window_ptr->draw(ESC_large);
544
545                     if (alpha < 255) {
546
547                         alpha += dalpha;
548
549                         if (alpha > 255) {
550                             alpha = 255;
551                         }
552
553                         ESC_large.setColor(sf::Color(255, 255, 255, alpha));
554                     }
555
556                     else {
557                         brand_state++;
558                     }
```

```
559
560                     break;
561                 }
562
563
564                 case (1): {
565                     // key press
566                     render_window_ptr->draw(ESC_small);
567
568                     if (click_frame < FRAMES_PER_SECOND / 8) {
569                         if (not sound_played) {
570                             brand_assets_manager.getSound("key press")->play();
571                             sound_played = true;
572                         }
573
574                         click_frame++;
575                     }
576
577                     else {
578                         brand_state++;
579                     }
580
581                     break;
582                 }
583
584
585                 case (2): {
586                     // text wave
587                     brand_text.setString(brand_string.substr(0, substring_idx));
588
589                     render_window_ptr->draw(brand_text);
590                     render_window_ptr->draw(ESC_large);
591
592                     if (substring_idx <= brand_string.size()) {
593                         if (brand_frame % (FRAMES_PER_SECOND / 20) == 0) {
594                             substring_idx++;
595                         }
596                     }
597
598                     else {
599                         brand_state++;
600                     }
601
602                     break;
603                 }
604
605
606                 case (3): {
607                     // fade out brand
608                     render_window_ptr->draw(brand_text);
609                     render_window_ptr->draw(ESC_large);
610
611                     if (alpha > 0) {
612                         alpha -= dalpha;
613
614                         if (alpha < 0) {
615                             alpha = 0;
616                         }
617
618                         brand_text.setFillColor(sf::Color(255, 255, 255, alpha));
619                         ESC_large.setColor(sf::Color(255, 255, 255, alpha));
620                     }
621
622                     else {
623                         brand_state++;
624                     }
625
626                     break;
627                 }
628
629
630                 case (4): {
631                     // fade in SFML
632                     render_window_ptr->draw(SFML);
633
634                     if (alpha < 255) {
635                         alpha += dalpha;
636
637                         if (alpha > 255) {
638                             alpha = 255;
639                         }
640
641                         SFML.setColor(sf::Color(255, 255, 255, alpha));
642                     }
643
644                     else {
645                         brand_state++;
```

```
646                     }
647
648                     break;
649                 }
650
651
652                 case (5): {
653                     // fade out SFML
654                     render_window_ptr->draw(SFML);
655
656                     if (alpha > 0) {
657                         alpha -= dalpha;
658
659                         if (alpha < 0) {
660                             alpha = 0;
661                         }
662
663                         SFML.setColor(sf::Color(255, 255, 255, alpha));
664                     }
665
666                     else {
667                         brand_state++;
668                     }
669
670                     break;
671                 }
672
673
674                 default: {
675                     // do nothing!
676
677                     break;
678                 }
679             }
680
681             render_window_ptr->display();
682             brand_frame++;
683         }
684     }
685
686     return;
687 }   /* playBrandAnimation() */
```

5.28.2.5 showTitleScreen()

```
void showTitleScreen (
              sf::RenderWindow * render_window_ptr,
              AssetsManager * assets_manager_ptr )
```

Helper function.

Parameters

render_window_ptr	A pointer to the render window.
assets_manager_ptr	A pointer to the assets manager.

```
712 {
713     //  1. set up and position title assets
714     int outline_thickness = 32;
715
716     sf::RectangleShape title_console(
717         sf::Vector2f(
718             GAME_WIDTH - 2 * outline_thickness,
719             GAME_HEIGHT - 2 * outline_thickness
720         )
721     );
722
723     title_console.setPosition(outline_thickness, outline_thickness);
724
725     sf::Color title_fill_colour = MONOCHROME_SCREEN_BACKGROUND;
726     title_fill_colour.a = 0;
727
728     sf::Color title_outline_colour = MENU_FRAME_GREY;
729     title_outline_colour.a = 0;
```

```
730
731     title_console.setFillColor(title_fill_colour);
732     title_console.setOutlineColor(title_outline_colour);
733     title_console.setOutlineThickness(outline_thickness);
734
735     std::string title_string_upper = "ROAD TO ZERO";
736     sf::Text title_text_upper(
737         title_string_upper,
738         *(assets_manager_ptr->getFont("Glass_TTY_VT220")),
739         128
740     );
741
742     title_text_upper.setOrigin(
743         title_text_upper.getLocalBounds().width / 2,
744         title_text_upper.getLocalBounds().height / 2
745     );
746
747     title_text_upper.setPosition(GAME_WIDTH / 2, GAME_HEIGHT / 2 - 128);
748     title_text_upper.setFillColor(MONOCHROME_TEXT_GREEN);
749
750     std::string title_string_lower = "THE MICROGRID MANAGEMENT GAME";
751     sf::Text title_text_lower(
752         title_string_lower,
753         *(assets_manager_ptr->getFont("Glass_TTY_VT220")),
754         64
755     );
756
757     title_text_lower.setOrigin(
758         title_text_lower.getLocalBounds().width / 2,
759         title_text_lower.getLocalBounds().height / 2
760     );
761
762     title_text_lower.setPosition(GAME_WIDTH / 2, GAME_HEIGHT / 2);
763     title_text_lower.setFillColor(MONOCHROME_TEXT_GREEN);
764
765     std::string title_string_bottom = "ROAD TO ZERO V";
766     title_string_bottom += GAME_VERSION;
767     title_string_bottom += "    COPYRIGHT 2023 - [ESC] INTERACTIVE";
768     sf::Text title_text_bottom(
769         title_string_bottom,
770         *(assets_manager_ptr->getFont("Glass_TTY_VT220")),
771         16
772     );
773
774     title_text_bottom.setOrigin(
775         title_text_bottom.getLocalBounds().width / 2,
776         title_text_bottom.getLocalBounds().height / 2
777     );
778
779     title_text_bottom.setPosition(GAME_WIDTH / 2, GAME_HEIGHT - 64);
780     title_text_bottom.setFillColor(MONOCHROME_TEXT_GREEN);
781
782     sf::Text prompt_text(
783         "PRESS ANY KEY TO CONTINUE",
784         *(assets_manager_ptr->getFont("Glass_TTY_VT220")),
785         24
786     );
787
788     prompt_text.setOrigin(
789         prompt_text.getLocalBounds().width / 2,
790         prompt_text.getLocalBounds().height / 2
791     );
792
793     prompt_text.setPosition(GAME_WIDTH / 2, GAME_HEIGHT / 2 + 175);
794     prompt_text.setFillColor(MONOCHROME_TEXT_GREEN);
795
796     sf::RectangleShape fade_rectangle(sf::Vector2f(GAME_WIDTH, GAME_HEIGHT));
797     sf::Color fade_rectangle_colour(0, 0, 0, 0);
798     fade_rectangle.setFillColor(fade_rectangle_colour);
799
800
801     //  2. draw loop
802     bool draw_title = true;
803     bool sound_played = false;
804
805     int title_frame = 0;
806     int title_state = 0;
807
808     size_t upper_substring_idx = 0;
809     size_t lower_substring_idx = 0;
810     size_t bottom_substring_idx = 0;
811
812     double alpha = 0;
813     double dalpha = FRAMES_PER_SECOND / 18;
814     double time_since_start_s = 0;
815
816     sf::Clock title_clock;
```

```
817     sf::Event title_event;
818 
819     while (draw_title) {
820         time_since_start_s = title_clock.getElapsedTime().asSeconds();
821 
822         if (time_since_start_s >= (title_frame + 1) * SECONDS_PER_FRAME) {
823             render_window_ptr->clear();
824 
825             //  2.1. title state switch
826             switch (title_state) {
827                 case (0): {
828                     while (render_window_ptr->pollEvent(title_event)) {
829                         if (title_event.type == sf::Event::Closed) {
830                             render_window_ptr->close();
831                             return;
832                         }
833                     }
834 
835                     // fade in title console
836                     render_window_ptr->draw(title_console);
837 
838                     if (alpha < 255) {
839                         alpha += dalpha;
840 
841                         if (alpha > 255) {
842                             alpha = 255;
843                         }
844 
845                         title_fill_colour.a = alpha;
846                         title_outline_colour.a = alpha;
847 
848                         title_console.setFillColor(title_fill_colour);
849                         title_console.setOutlineColor(title_outline_colour);
850                     }
851 
852                     else {
853                         title_state++;
854                         alpha = 0;
855                     }
856 
857                     break;
858                 }
859 
860 
861                 case (1): {
862                     while (render_window_ptr->pollEvent(title_event)) {
863                         if (title_event.type == sf::Event::Closed) {
864                             render_window_ptr->close();
865                             return;
866                         }
867                     }
868 
869                     // fade in title text
870                     if (not sound_played) {
871                         assets_manager_ptr->getSound("game title screen")->play();
872                         sound_played = true;
873                     }
874 
875                     if (title_string_bottom.substr(0, bottom_substring_idx) != title_string_bottom) {
876                         title_text_upper.setString(title_string_upper.substr(0, upper_substring_idx));
877                         title_text_lower.setString(title_string_lower.substr(0, lower_substring_idx));
878                         title_text_bottom.setString(title_string_bottom.substr(0,
    bottom_substring_idx));
879 
880                         assets_manager_ptr->getSound("console string print")->play();
881 
882                         upper_substring_idx++;
883                         lower_substring_idx++;
884                         bottom_substring_idx++;
885 
886                         if (upper_substring_idx > title_string_upper.size()) {
887                             upper_substring_idx = title_string_upper.size();
888                         }
889 
890                         if (lower_substring_idx > title_string_lower.size()) {
891                             lower_substring_idx = title_string_lower.size();
892                         }
893                     }
894 
895                     else {
896                         title_text_upper.setString(title_string_upper.substr(0, upper_substring_idx));
897                         title_text_lower.setString(title_string_lower.substr(0, lower_substring_idx));
898                         title_text_bottom.setString(title_string_bottom.substr(0,
    bottom_substring_idx));
899                         title_state++;
900                     }
901 
```

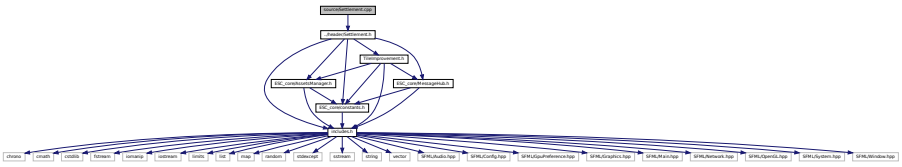
```
902                     render_window_ptr->draw(title_console);
903                     render_window_ptr->draw(title_text_upper);
904                     render_window_ptr->draw(title_text_lower);
905                     render_window_ptr->draw(title_text_bottom);
906 
907                     break;
908                 }
909 
910 
911                 case (2): {
912                     while (render_window_ptr->pollEvent(title_event)) {
913                         if (title_event.type == sf::Event::KeyPressed) {
914                             title_state++;
915                         }
916 
917                         if (title_event.type == sf::Event::Closed) {
918                             render_window_ptr->close();
919                             return;
920                         }
921                     }
922 
923                     // flashing prompt
924                     render_window_ptr->draw(title_console);
925                     render_window_ptr->draw(title_text_upper);
926                     render_window_ptr->draw(title_text_lower);
927                     render_window_ptr->draw(title_text_bottom);
928 
929                     if (
930                         (title_frame > 3.5 * FRAMES_PER_SECOND) and
931                         ((title_frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2)
932                     ) {
933                         render_window_ptr->draw(prompt_text);
934                     }
935 
936                     break;
937                 }
938 
939 
940                 case (3): {
941                     while (render_window_ptr->pollEvent(title_event)) {
942                         if (title_event.type == sf::Event::Closed) {
943                             render_window_ptr->close();
944                             return;
945                         }
946                     }
947 
948                     // fade out
949                     render_window_ptr->draw(title_console);
950                     render_window_ptr->draw(title_text_upper);
951                     render_window_ptr->draw(title_text_lower);
952                     render_window_ptr->draw(title_text_bottom);
953 
954                     if ((title_frame % FRAMES_PER_SECOND) > FRAMES_PER_SECOND / 2) {
955                         render_window_ptr->draw(prompt_text);
956                     }
957 
958                     render_window_ptr->draw(fade_rectangle);
959 
960                     if (alpha < 255) {
961                         alpha += dalpha;
962 
963                         if (alpha > 255) {
964                             alpha = 255;
965                         }
966 
967                         fade_rectangle_colour.a = alpha;
968 
969                         fade_rectangle.setFillColor(fade_rectangle_colour);
970                     }
971 
972                     else {
973                         draw_title = false;
974                     }
975 
976                     break;
977                 }
978 
979 
980                 default: {
981                     // do nothing!
982 
983                     break;
984                 }
985             }
986 
987             render_window_ptr->display();
988             title_frame++;
```

```
989         }
990     }
991
992     return;
993 }   /* showTitleScreen() */
```

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

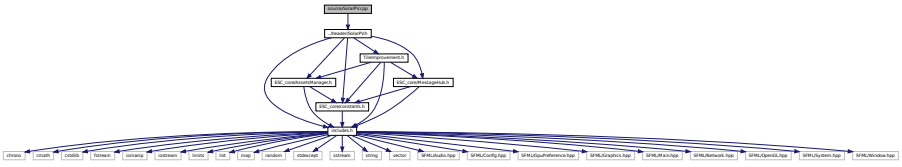
Implementation file for the Settlement class.

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.

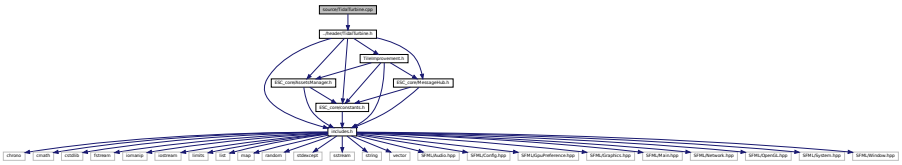
A base class for the tile improvement hierarchy.

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.

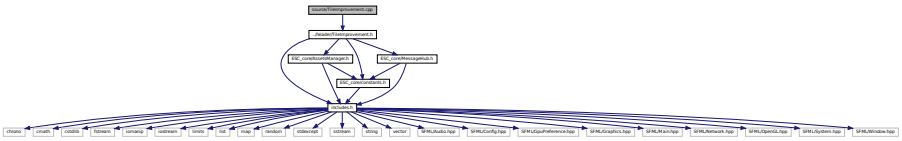
A base class for the tile improvement hierarchy.

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.

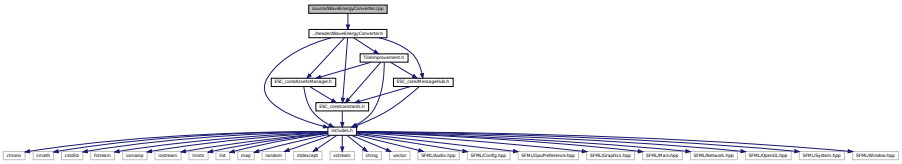
A base class for the tile improvement hierarchy.

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.

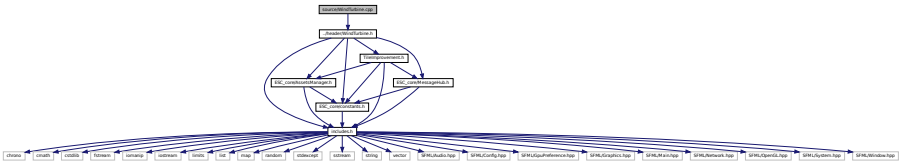
A base class for the tile improvement hierarchy.

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/. 310

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 309

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 40, 54, 128, 181, 191, 208, 228, 251, 270

Index