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:

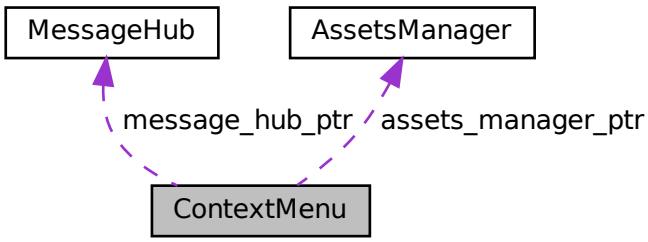
- header/ESC_core/AssetsManager.h
- source/ESC_core/AssetsManager.cpp

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:

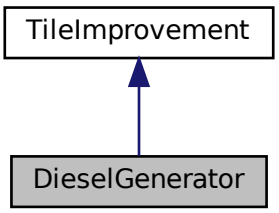
- header/ContextMenu.h
- source/ContextMenu.cpp

4.3 DieselGenerator Class Reference

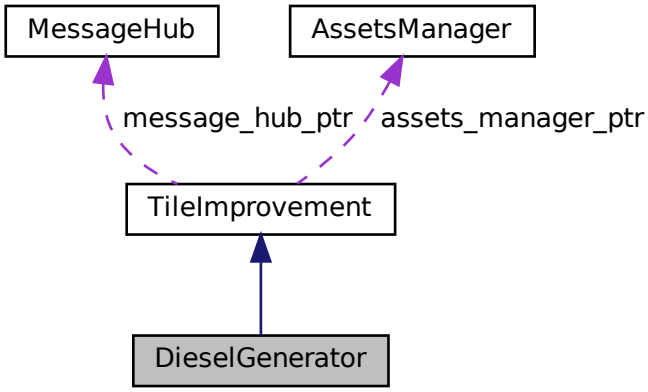
A settlement class (child class of TileImprovement).

```
#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.

```
502   :
503 TileImprovement(
504     position_x,
505     position_y,
506     tile_resource,
507     event_ptr,
508     render_window_ptr,
509     assets_manager_ptr,
510     message_hub_ptr
511 )
512 {
513     //  1. set attributes
514
515     //  1.1. private
516     //...
517
518     //  1.2. public
519     this->tile_improvement_type = TileImprovementType :: DIESEL_GENERATOR;
520
521     this->is_running = false;
522
523     this->health = 100;
524
525     this->capacity_kW = 200;
526     this->upgrade_level = 1;
527
528     this->production_MWh = 0;
529     this->max_production_MWh = 144;
530
531     this->smoke_da = 1e-8 * SECONDS_PER_FRAME;
532     this->smoke_dx = 5 * SECONDS_PER_FRAME;
533     this->smoke_dy = -10 * SECONDS_PER_FRAME;
534     this->smoke_prob = 16 * SECONDS_PER_FRAME;
535
536     this->smoke_sprite_list = {};
537
538     this->fuel_cost = 0;
539     this->emissions_tonnes_CO2e = 0;
540
541     this->tile_improvement_string = "DIESEL GEN";
542
543     this->__setUpTileImprovementSpriteAnimated();
544
545     std::cout « "DieselGenerator constructed at " « this « std::endl;
546
547     return;
```

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

4.3.2.2 ~DieselGenerator()

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

Destructor for the DieselGenerator class.

```
929 {
930     std::cout « "DieselGenerator at " « this « " destroyed" « std::endl;
931
932     return;
933 }   /* ~DieselGenerator() */
```

4.3.3 Member Function Documentation

4.3.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.3.3.2 __computeProductionCosts()

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

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

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

4.3.3.3 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.3.3.4 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

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

4.3.3.5 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

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

4.3.3.6 __repair()

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

Helper method to repair the diesel generator.

Reimplemented from TileImprovement.

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

4.3.3.7 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

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

4.3.3.8 __setUpTileImprovementSpriteAnimated()

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

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

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

4.3.3.9 __upgrade()

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

Helper method to upgrade the diesel generator.

```
185 {
186     if (this->credits < DIESEL_GENERATOR_BUILD_COST) {
187         std::cout « "Cannot upgrade diesel generator: insufficient credits (need "
188             « DIESEL_GENERATOR_BUILD_COST « " K)" « std::endl;
189
190         this->__sendInsufficientCreditsMessage();
191         return;
192     }
193
194     if (this->upgrade_level >= MAX_UPGRADE_LEVELS) {
195         return;
196     }
197
198     this->is_running = false;
199
200     TileImprovement :: __repair();
201
202     this->capacity_kW += 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.

```
658 {
659     //  1. send improvement state message
660     this->__sendImprovementStateMessage();
661
662     //  2. handle start/stop
663     if ((not this->is_running) and (this->production_MWh > 0)) {
664         this->is_running = true;
665         this->assets_manager_ptr->getSound("diesel start")->play();
666     }
667
668     else if (this->is_running and (this->production_MWh <= 0)) {
669         this->is_running = false;
670         this->tile_improvement_sprite_animated[1].setScale(sf::Vector2f(1, 1));
671     }
672
673     //  3. handle equipment health and breakdowns
674     if (this->is_running) {
675         this->health--;
676
677         if (this->health <= 50) {
678             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
679
680             if ((double)rand() / RAND_MAX <= breakdown_prob) {
681                 this->health = 0;
682             }
683         }
684
685         if (this->health <= 0) {
686             this->__breakdown();
687         }
688     }
689
690     //  4. send tile state request (if selected)
691     if (this->is_selected) {
692         this->__sendTileStateRequest();
693     }
694
695     return;
696 }   /* 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.

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

```
777                 )
778             );
779 
780             this->tile_improvement_sprite_animated[i].setScale(
781                 sf::Vector2f(
782                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
783                     1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
784                 )
785             );
786         }
787 
788         this->upgrade_frame++;
789     }
790 
791     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
792         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
793             this->tile_improvement_sprite_animated[i].setColor(
794                 sf::Color(255,255,255,255)
795             );
796 
797             this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
798         }
799 
800         this->just_upgraded = false;
801         this->upgrade_frame = 0;
802     }
803 
804 
805     //  3. draw first element of animated sprite
806     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
807 
808 
809     //  4. draw second element of animated sprite
810     double move_x = 0;
811     double move_y = 0;
812 
813     if (this->is_running) {
814         this->tile_improvement_sprite_animated[1].setScale(
815             sf::Vector2f(
816                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2),
817                 1 + 0.05 * pow(cos((6 * M_PI * this->frame) / FRAMES_PER_SECOND), 2)
818             )
819         );
820 
821         move_x = 1 * ((double)rand() / RAND_MAX) - 0.5;
822         move_y = 1 * ((double)rand() / RAND_MAX) - 0.5;
823 
824         this->tile_improvement_sprite_animated[1].move(move_x, move_y);
825     }
826 
827     this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
828 
829     if (this->is_running) {
830         this->tile_improvement_sprite_animated[1].move(-1 * move_x, -1 * move_y);
831     }
832 
833 
834     //  5. draw smoke effects
835     if (this->is_running) {
836         if ((double)rand() / RAND_MAX < smoke_prob) {
837             this->smoke_sprite_list.push_back(
838                 sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
839             );
840 
841             this->smoke_sprite_list.back().setOrigin(
842                 this->smoke_sprite_list.back().getLocalBounds().width / 2,
843                 this->smoke_sprite_list.back().getLocalBounds().height / 2
844             );
845 
846             this->smoke_sprite_list.back().setPosition(
847                 this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
848                 this->position_y - 33
849             );
850         }
851     }
852 
853     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
854 
855     double alpha = 255;
856 
857     while (iter != this->smoke_sprite_list.end()) {
858         this->render_window_ptr->draw(*iter);
859 
860         alpha = (*iter).getColor().a;
861 
862         alpha -= this->smoke_da;
863 
```

```
864         if (alpha <= 0) {
865             iter = this->smoke_sprite_list.erase(iter);
866             continue;
867         }
868 
869         (*iter).setColor(sf::Color(255, 255, 255, alpha));
870 
871         (*iter).move(
872             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
873             this->smoke_dy
874         );
875 
876         (*iter).rotate(((double)rand() / RAND_MAX));
877 
878         iter++;
879     }
880 
881 
882     //  6. handle dispatch illustration
883     if (this->production_MWh > 0) {
884         this->dispatch_text.setString(std::to_string(this->production_MWh));
885         this->__drawDispatch();
886     }
887 
888 
889     //  7. draw production menu
890     if (this->production_menu_open) {
891         this->render_window_ptr->draw(this->production_menu_backing);
892         this->render_window_ptr->draw(this->production_menu_backing_text);
893 
894         this->__drawProductionMenu();
895     }
896 
897 
898     //  8. handle broken effects
899     if (this->is_broken) {
900         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
901             this->tile_improvement_sprite_animated[i].setColor(
902                 sf::Color(
903                     255,
904                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
905                     255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
906                     255
907                 )
908             );
909         }
910     }
911 
912     this->frame++;
913     return;
914 }   /* draw() */
```

4.3.3.12 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

```
565 {
566     int upgrade_cost = DIESEL_GENERATOR_BUILD_COST;
567 
568     //                  32 char x 17 line console "--------------------------------\n";
569     std::string options_substring                = "CAPACITY:    ";
570     options_substring                           += std::to_string(this->capacity_kW);
571     options_substring                           += " kW (level ";
572     options_substring                           += std::to_string(this->upgrade_level);
573     options_substring                           += ")\n";
574 
```

```
575     options_substring                           += "PRODUCTION:  ";
576     options_substring                           += std::to_string(this->production_MWh);
577     options_substring                           += " MWh (MAX ";
578     options_substring                           += std::to_string(this->max_production_MWh);
579     options_substring                           += ")\n";
580
581     options_substring                           += "HEALTH:      ";
582     options_substring                           += std::to_string(this->health);
583     options_substring                           += "/100";
584
585     if (this->health <= 0) {
586         options_substring                       += " ** BROKEN! **\n";
587     }
588
589     else {
590         options_substring                       += "\n";
591     }
592
593     options_substring                           += "                                \n";
594     options_substring                           += "  **** DIESEL GEN OPTIONS ****  \n";
595     options_substring                           += "                                \n";
596
597     if (this->is_broken) {
598         options_substring                       += "     [R]:  REPAIR (";
599         options_substring                       += std::to_string(DIESEL_GENERATOR_BUILD_COST);
600         options_substring                       += " K)\n";
601     }
602
603     else {
604         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
605     }
606
607     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
608         options_substring                           += "     [U]:  + 200 kW (";
609         options_substring                           += std::to_string(upgrade_cost);
610         options_substring                           +=" K)\n";
611     }
612
613     options_substring                           += "HOLD [P]:  SCRAP (";
614     options_substring                           += std::to_string(SCRAP_COST);
615     options_substring                           += " K)";
616
617     return options_substring;
618 }   /* getTileOptionsSubstring() */
```

4.3.3.13 processEvent()

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

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

Reimplemented from TileImprovement.

```
711 {
712     TileImprovement :: processEvent();
713
714     if (this->event_ptr->type == sf::Event::KeyPressed) {
715         this->__handleKeyPressEvents();
716     }
717
718     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
719         this->__handleMouseButtonEvents();
720     }
721
722     return;
723 }   /* processEvent() */
```

4.3.3.14 processMessage()

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

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

Reimplemented from TileImprovement.

```
738 {
739     TileImprovement :: processMessage();
740
741     //...
742
743     return;
744 }   /* processMessage() */
```

4.3.3.15 setIsSelected()

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

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

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

4.3.4 Member Data Documentation

4.3.4.1 capacity_kW

```
int DieselGenerator::capacity_kW
```

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

4.3.4.2 emissions_tonnes_CO2e

```
int DieselGenerator::emissions_tonnes_CO2e
```

The emissions for this turn.

4.3.4.3 fuel_cost

```
int DieselGenerator::fuel_cost
```

The fuel costs for this turn.

4.3.4.4 max_production_MWh

```
int DieselGenerator::max_production_MWh
```

The maximum production [MWh] for this turn.

4.3.4.5 production_MWh

```
int DieselGenerator::production_MWh
```

The current production [MWh] of the diesel generator.

4.3.4.6 smoke_da

```
double DieselGenerator::smoke_da
```

The per frame delta in smoke particle alpha value.

4.3.4.7 smoke_dx

```
double DieselGenerator::smoke_dx
```

The per frame delta in smoke particle x position.

4.3.4.8 smoke_dy

```
double DieselGenerator::smoke_dy
```

The per frame delta in smoke particle y position.

4.3.4.9 smoke_prob

```
double DieselGenerator::smoke_prob
```

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

4.3.4.10 smoke_sprite_list

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

A list of smoke sprite (for exhaust animation).

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

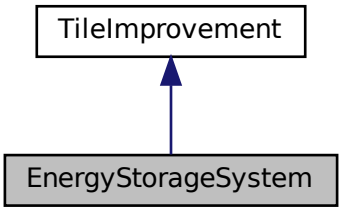
- header/DieselGenerator.h
- source/DieselGenerator.cpp

4.4 EnergyStorageSystem Class Reference

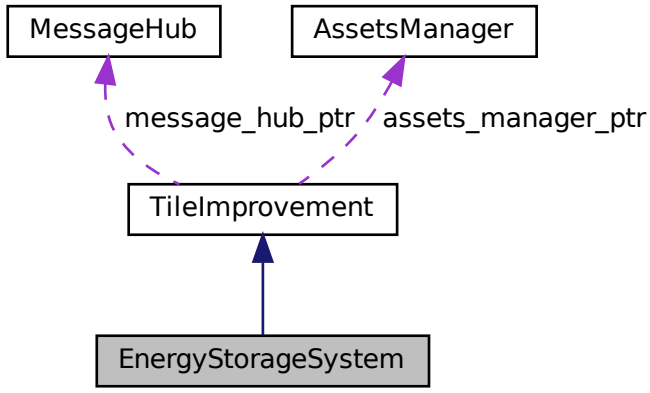
A settlement class (child class of TileImprovement).

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

Inheritance diagram for EnergyStorageSystem:



Collaboration diagram for EnergyStorageSystem:



Public Member Functions

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

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

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

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

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

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

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

  Destructor for the EnergyStorageSystem class.

Public Attributes

- int capacity_MWh

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

  The charge [MWh] in the energy storage system.

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

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

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.4.1 Detailed Description

A settlement class (child class of TileImprovement).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 EnergyStorageSystem()

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

Constructor for the EnergyStorageSystem class.

Ref: Wikipedia [2023]

Parameters

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

```
291   :
292 TileImprovement(
```

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

4.4.2.2 ~EnergyStorageSystem()

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

Destructor for the EnergyStorageSystem class.

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

4.4.3 Member Function Documentation

4.4.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

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

4.4.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.4.3.3 __setUpProductionMenu()

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

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

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

4.4.3.4 __setUpTileImprovementSpriteStatic()

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

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

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

4.4.3.5 __upgrade()

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

Helper method to upgrade the diesel generator.

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

4.4.3.6 draw()

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

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

Reimplemented from TileImprovement.

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

4.4.3.7 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

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

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

4.4.3.8 processEvent()

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

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

Reimplemented from TileImprovement.

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

4.4.3.9 processMessage()

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

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

Reimplemented from TileImprovement.

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

4.4.3.10 setIsSelected()

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

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

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

4.4.4 Member Data Documentation

4.4.4.1 capacity_MWh

```
int EnergyStorageSystem::capacity_MWh
```

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

4.4.4.2 charge_MWh

```
int EnergyStorageSystem::charge_MWh
```

The charge [MWh] in the energy storage system.

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

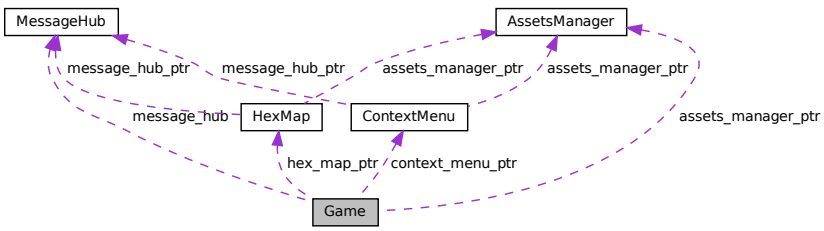
- header/EnergyStorageSystem.h
- source/EnergyStorageSystem.cpp

4.5 Game Class Reference

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

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

Collaboration diagram for Game:



Public Member Functions

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

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

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

  Destructor for the Game class.

Public Attributes

- GamePhase game_phase

  The current phase of the game.
- bool quit_game

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

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

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

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

  A boolean indicating whether a message deadlock has been detected.
- bool show_tutorial

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

  A boolean indicating a turn end.
- 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.
- 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.
- 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 )
```

Constructor for the Game class.

```
1723 {
1724     //  1. set attributes
1725
1726     //  1.1. private
1727     this->render_window_ptr = render_window_ptr;
1728
1729     this->assets_manager_ptr = assets_manager_ptr;
1730
1731     //  1.2. public
1732     this->game_phase = GamePhase :: BUILD_SETTLEMENT;
1733
1734     this->quit_game = false;
1735     this->game_loop_broken = false;
1736     this->show_frame_clock_overlay = false;
1737     this->check_terminating_conditions = false;
1738     this->show_tutorial = true;
1739     this->turn_end = false;
1740     this->draw_turn_advance_banner = false;
1741     this->increase_turn_advance_alpha = true;
1742
1743     this->tutorial_page = 0;
1744     this->tutorial_string = TUTORIAL_PAGES[this->tutorial_page];
1745
1746     this->tutorial_text.setFont(
1747         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
1748     );
1749     this->tutorial_text.setCharacterSize(16);
1750     this->tutorial_text.setFillColor(MONOCHROME_TEXT_GREEN);
1751     this->tutorial_text.setPosition(GAME_WIDTH - 400 + 64, 64);
```

```
1752
1753     this->frame = 0;
1754     this->time_since_start_s = 0;
1755
1756     this->message_deadlock = false;
1757     this->message_deadlock_frame = 0;
1758
1759     double seconds_since_epoch = time(NULL);
1760     double years_since_epoch = seconds_since_epoch / SECONDS_PER_YEAR;
1761
1762     this->year = 1970 + (int)years_since_epoch;
1763     this->month = 0;
1764
1765     this->population = 0;
1766     this->credits = STARTING_CREDITS;
1767     this->demand_MWh = 0;
1768     this->cumulative_emissions_tonnes = 0;
1769
1770     this->past_demand_MWh = 0;
1771     this->turn_advance_alpha = 0;
1772
1773     this->demand_vec_MWh.resize(30, 0);
1774
1775     this->demand_served_MWh = 0;
1776     this->demand_remaining_MWh = 0;
1777     this->overproduction_MWh = 0;
1778     this->turn_fuel_cost = 0;
1779     this->turn_operation_maintenance_cost = 0;
1780     this->turn_emissions_tonnes = 0;
1781
1782     this->overproduction_penalty = 0;
1783     this->dispatch_income = 0;
1784     this->net_credit_flow = 0;
1785
1786     this->consecutive_zero_emissions_months = 0;
1787
1788     this->substring_idx = 0;
1789     this->turn_summary_string = "";
1790
1791     this->turn_summary_text.setFont(
1792         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
1793     );
1794     this->turn_summary_text.setCharacterSize(16);
1795     this->turn_summary_text.setFillColor(MONOCHROME_TEXT_GREEN);
1796     this->turn_summary_text.setPosition(GAME_WIDTH - 400 + 64, 64);
1797
1798     this->hex_map_ptr = new HexMap(
1799         6,
1800         &(this->event),
1801         this->render_window_ptr,
1802         this->assets_manager_ptr,
1803         &(this->message_hub)
1804     );
1805
1806     this->context_menu_ptr = new ContextMenu(
1807         &(this->event),
1808         this->render_window_ptr,
1809         this->assets_manager_ptr,
1810         &(this->message_hub)
1811     );
1812
1813     //  2. add message channel(s)
1814     this->message_hub.addChannel(GAME_CHANNEL);
1815     this->message_hub.addChannel(GAME_STATE_CHANNEL);
1816
1817     this->__sendGameStateMessage();
1818
1819     std::cout « "Game constructed at " « this « std::endl;
1820
1821     return;
1822 }   /* Game() */
```

4.5.2.2 ~Game()

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

Destructor for the Game class.

```
1946 {
1947     //  1. clean up attributes
1948     delete this->hex_map_ptr;
1949     delete this->context_menu_ptr;
1950 
1951     std::cout « "Game at " « this « " destroyed" « std::endl;
1952 
1953     return;
1954 }   /* ~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. terminating conditions
1661     switch (this->game_phase) {
1662         case (GamePhase :: LOSS_DEMAND): {
1663             this->__drawLossDemand();
1664 
1665             break;
1666         }
1667 
1668 
1669         case (GamePhase :: LOSS_CREDITS): {
1670             this->__drawLossCredits();
1671 
1672             break;
1673         }
1674 
1675 
1676         case (GamePhase :: LOSS_EMISSIONS): {
1677             this->__drawLossEmissions();
1678 
1679             break;
1680         }
1681 
1682 
1683         case (GamePhase ::VICTORY): {
1684             this->__drawVictory();
1685 
1686             break;
1687         }
1688 
1689 
1690         default: {
```

```
1691             // do nothing!
1692
1693             break;
1694         }
1695     }
1696
1697     return;
1698 }   /* 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).

```
1840 {
1841     // start game loop
1842     while (not this->game_loop_broken) {
1843         this->time_since_start_s = this->clock.getElapsedTime().asSeconds();
1844 
1845         if (this->time_since_start_s >= (this->frame + 1) * SECONDS_PER_FRAME) {
1846             //  process events
1847             while (this->render_window_ptr->pollEvent(this->event)) {
1848                 if (
1849                     (this->game_phase == GamePhase :: BUILD_SETTLEMENT) or
1850                     (this->game_phase == GamePhase :: SYSTEM_MANAGEMENT)
1851                 ) {
1852                     this->hex_map_ptr->processEvent();
1853                     this->context_menu_ptr->processEvent();
1854                     this->__processEvent();
1855                 }
1856 
1857                 else {
1858                     if (this->event.type == sf::Event::KeyPressed) {
1859                         this->game_loop_broken = true;
1860                     }
1861                 }
1862             }
1863 
1864 
1865             //  process messages
1866             while (this->message_hub.hasTraffic()) {
1867                 this->hex_map_ptr->processMessage();
1868                 this->context_menu_ptr->processMessage();
1869                 this->__processMessage();
1870 
1871                 this->check_terminating_conditions = true;
1872 
1873                 if (not this->message_deadlock) {
1874                     this->message_deadlock_frame++;
1875 
1876                     if (this->message_deadlock_frame > 5 * FRAMES_PER_SECOND) {
1877                         this->message_hub.printState();
1878                         this->message_deadlock = true;
1879                     }
1880                 }
1881             }
1882             this->message_deadlock = false;
1883             this->message_deadlock_frame = 0;
1884 
1885 
1886             //  handle turn end summary
1887             if (this->turn_end) {
1888                 std::cout « "**** END OF TURN " « std::to_string(this->turn - 1) «
1889                     " ****" « std::endl;
1890 
1891                 this->__summarizeTurn();
1892 
1893                 this->turn_end = false;
1894 
1895                 this->draw_turn_advance_banner = true;
1896                 this->turn_advance_alpha = 0;
1897                 this->increase_turn_advance_alpha = true;
1898             }
1899 
1900 
1901             //  check terminating conditions
1902             if (this->check_terminating_conditions) {
1903                 this->__checkTerminatingConditions();
1904                 this->check_terminating_conditions = false;
1905             }
1906 
1907 
1908             //  draw frame
1909             this->render_window_ptr->clear();
1910 
1911             this->hex_map_ptr->draw();
1912             this->context_menu_ptr->draw();
1913             this->__draw();
1914 
1915             this->render_window_ptr->display();
1916 
1917 
1918             //  increment frame
1919             this->frame++;
1920         }
1921 
1922         // check track status, move to next if stopped
1923         if (this->assets_manager_ptr->getTrackStatus() == sf::SoundSource::Stopped) {
```

```
1924             this->assets_manager_ptr->nextTrack();
1925             this->assets_manager_ptr->playTrack();
1926         }
1927
1928     }
1929
1930     return this->quit_game;
1931 }   /* 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 frame

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

The current frame of the game.

4.5.4.16 game_loop_broken

```
bool Game::game_loop_broken
```

Boolean indicating whether or not the game loop is broken.

4.5.4.17 game_phase

```
GamePhase Game::game_phase
```

The current phase of the game.

4.5.4.18 hex_map_ptr

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

Pointer to the hex map (defines game world).

4.5.4.19 increase_turn_advance_alpha

```
bool Game::increase_turn_advance_alpha
```

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

4.5.4.20 message_deadlock

```
bool Game::message_deadlock
```

A boolean indicating whether a message deadlock has been detected.

4.5.4.21 message_deadlock_frame

```
int Game::message_deadlock_frame
```

A frame counter for detecting message deadlock.

4.5.4.22 message_hub

```
MessageHub Game::message_hub
```

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

4.5.4.23 month

```
int Game::month
```

Current game month.

4.5.4.24 net_credit_flow

```
int Game::net_credit_flow
```

The net credit flow at the end of a turn.

4.5.4.25 overproduction_MWh

```
int Game::overproduction_MWh
```

The amount of overproduction at the end of a turn.

4.5.4.26 overproduction_penalty

```
int Game::overproduction_penalty
```

The penalty for overproduction.

4.5.4.27 past_demand_MWh

```
int Game::past_demand_MWh
```

The demand in the previous turn.

4.5.4.28 population

```
int Game::population
```

Current population.

4.5.4.29 quit_game

```
bool Game::quit_game
```

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

4.5.4.30 render_window_ptr

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

A pointer to the render window.

4.5.4.31 show_frame_clock_overlay

```
bool Game::show_frame_clock_overlay
```

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

4.5.4.32 show_tutorial

```
bool Game::show_tutorial
```

A boolean indicating whether or not to show the tutorial.

4.5.4.33 substring_idx

```
size_t Game::substring_idx
```

The index of the turn summary or tutorial substring.

4.5.4.34 time_since_start_s

```
double Game::time_since_start_s
```

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

4.5.4.35 turn

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

The current game turn.

4.5.4.36 turn_advance_alpha

```
double Game::turn_advance_alpha
```

The alpha value for the turn advance banner.

4.5.4.37 turn_emissions_tonnes

```
int Game::turn_emissions_tonnes
```

The amount of emissions at the end of a turn.

4.5.4.38 turn_end

```
bool Game::turn_end
```

A boolean indicating a turn end.

4.5.4.39 turn_fuel_cost

```
int Game::turn_fuel_cost
```

The cost of fuel at the end of a turn.

4.5.4.40 turn_operation_maintenance_cost

```
int Game::turn_operation_maintenance_cost
```

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

4.5.4.41 turn_summary_string

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

A string representation of the end of turn summary.

4.5.4.42 turn_summary_text

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

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

4.5.4.43 tutorial_page

```
size_t Game::tutorial_page
```

Index for which page of the tutorial to show.

4.5.4.44 tutorial_string

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

A string representation of the current tutorial page.

4.5.4.45 tutorial_text

```
sf::Text Game::tutorial_text
```

A text representation (drawable) of the tutorial page.

4.5.4.46 year

```
int Game::year
```

Current game year.

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

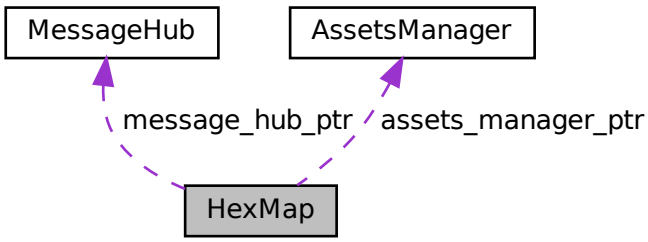
- header/Game.h
- source/Game.cpp

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:

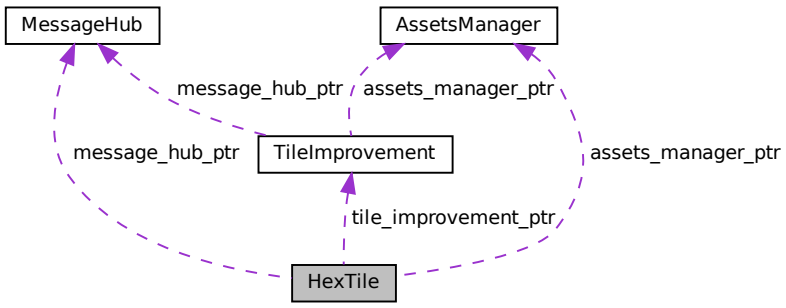
- header/HexMap.h
- source/HexMap.cpp

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.

```
2332 {
2333     //  1. set attributes
2334
2335     //  1.1. private
2336     this->event_ptr = event_ptr;
2337     this->render_window_ptr = render_window_ptr;
2338
2339     this->assets_manager_ptr = assets_manager_ptr;
2340     this->message_hub_ptr = message_hub_ptr;
2341
2342     //  1.2. public
2343     this->show_node = false;
2344     this->show_resource = false;
2345     this->resource_assessed = false;
2346     this->resource_assessment = false;
2347     this->is_selected = false;
2348     this->draw_explosion = false;
2349
2350     this->decoration_cleared = false;
2351     this->has_improvement = false;
2352     this->tile_improvement_ptr = NULL;
2353
2354     this->build_menu_open = false;
2355
2356     this->explosion_frame = 0;
2357
2358     this->frame = 0;
2359     this->credits = 0;
2360
2361     this->scrap_improvement_frame = 0;
2362
```

```
2363     this->position_x = position_x;
2364     this->position_y = position_y;
2365
2366     this->major_radius = 32;
2367     this->minor_radius = (sqrt(3) / 2) * this->major_radius;
2368
2369     this->game_phase = "build settlement";
2370
2371     //  2. set up and position drawable attributes
2372     this->__setUpNodeSprite();
2373     this->__setUpTileSprite();
2374     this->__setUpSelectOutlineSprite();
2375     this->__setUpResourceChipSprite();
2376     this->__setResourceText();
2377     this->__setUpMagnifyingGlassSprite();
2378     this->__setUpTileExplosionReel();
2379
2380     //  3. set tile type and resource (default to none type and average)
2381     this->setTileType(TileType :: NONE_TYPE);
2382     this->setTileResource(TileResource :: AVERAGE);
2383
2384     std::cout « "HexTile constructed at " « this « std::endl;
2385
2386     return;
2387 }   /* HexTile() */
```

4.7.2.2 ~HexTile()

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

Destructor for the HexTile class.

```
2955 {
2956     if (this->tile_improvement_ptr != NULL) {
2957         delete this->tile_improvement_ptr;
2958     }
2959
2960     std::cout « "HexTile at " « this « " destroyed" « std::endl;
2961
2962     return;
2963 }   /* ~HexTile() */
```

4.7.3 Member Function Documentation

4.7.3.1 __buildDieselGenerator()

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

Helper method to build a diesel generator on this tile.

```
1409 {
1410     int build_cost = DIESEL_GENERATOR_BUILD_COST;
1411
1412     if (this->credits < build_cost) {
1413         std::cout « "Cannot build diesel generator: insufficient credits (need "
1414             « build_cost « " K)" « std::endl;
1415
1416         this->__sendInsufficientCreditsMessage();
1417         return;
1418     }
1419
1420     this->tile_improvement_ptr = new DieselGenerator(
1421         this->position_x,
1422         this->position_y,
1423         this->tile_resource,
```

```
1424         this->event_ptr,
1425         this->render_window_ptr,
1426         this->assets_manager_ptr,
1427         this->message_hub_ptr
1428     );
1429
1430     this->has_improvement = true;
1431     this->__closeBuildMenu();
1432
1433     if (not this->resource_assessed) {
1434         this->assess();
1435     }
1436
1437     this->__sendCreditsSpentMessage(build_cost);
1438     this->__sendTileStateMessage();
1439     this->__sendGameStateRequest();
1440
1441     return;
1442 }   /* __buildDieselGenerator() */
```

4.7.3.2 __buildEnergyStorage()

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

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

```
1677 {
1678     /*
1679     int build_cost = ENERGY_STORAGE_SYSTEM_BUILD_COST;
1680
1681     if (this->credits < build_cost) {
1682         std::cout « "Cannot build energy storage system: insufficient credits (need "
1683             « build_cost « " K)" « std::endl;
1684
1685         this->__sendInsufficientCreditsMessage();
1686         return;
1687     }
1688
1689     this->tile_improvement_ptr = new EnergyStorageSystem(
1690         this->position_x,
1691         this->position_y,
1692         this->event_ptr,
1693         this->render_window_ptr,
1694         this->assets_manager_ptr,
1695         this->message_hub_ptr
1696     );
1697
1698     this->has_improvement = true;
1699     this->__closeBuildMenu();
1700
1701     if (not this->resource_assessed) {
1702         this->assess();
1703     }
1704
1705     this->__sendCreditsSpentMessage(build_cost);
1706     this->__sendTileStateMessage();
1707     this->__sendGameStateRequest();
1708     */
1709     return;
1710 }   /* __buildEnergyStorage() */
```

4.7.3.3 __buildSettlement()

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

Helper method to build a settlement on this tile.

```
1362 {
1363     if (this->credits < BUILD_SETTLEMENT_COST) {
```

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

4.7.3.4 __buildSolarPV()

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

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

```
1457 {
1458     int build_cost = SOLAR_PV_BUILD_COST;
1459
1460     if (this->tile_type == TileType :: LAKE) {
1461         build_cost *= SOLAR_PV_WATER_BUILD_MULTIPLIER;
1462     }
1463
1464     if (this->credits < build_cost) {
1465         std::cout « "Cannot build solar PV array: insufficient credits (need "
1466             « build_cost « " K)" « std::endl;
1467
1468         this->__sendInsufficientCreditsMessage();
1469         return;
1470     }
1471
1472     this->tile_improvement_ptr = new SolarPV(
1473         this->position_x,
1474         this->position_y,
1475         this->tile_resource,
1476         this->event_ptr,
1477         this->render_window_ptr,
1478         this->assets_manager_ptr,
1479         this->message_hub_ptr
1480     );
1481
1482     this->has_improvement = true;
1483     this->__closeBuildMenu();
1484
1485     if (not this->resource_assessed) {
1486         this->assess();
1487     }
1488
1489     if (this->tile_type == TileType :: LAKE) {
1490         this->decoration_cleared = true;
1491         this->assets_manager_ptr->getSound("splash")->play();
1492     }
1493
1494     this->__sendCreditsSpentMessage(build_cost);
1495     this->__sendTileStateMessage();
1496     this->__sendGameStateRequest();
```

```
1497
1498     return;
1499 }   /* __buildSolarPV() */
```

4.7.3.5 __buildTidalTurbine()

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

Helper method to build a tidal turbine on this tile.

```
1577 {
1578     int build_cost = TIDAL_TURBINE_BUILD_COST;
1579
1580     if (this->credits < build_cost) {
1581         std::cout « "Cannot build tidal turbine: insufficient credits (need "
1582             « build_cost « " K)" « std::endl;
1583
1584         this->__sendInsufficientCreditsMessage();
1585         return;
1586     }
1587
1588     this->tile_improvement_ptr = new TidalTurbine(
1589         this->position_x,
1590         this->position_y,
1591         this->tile_resource,
1592         this->event_ptr,
1593         this->render_window_ptr,
1594         this->assets_manager_ptr,
1595         this->message_hub_ptr
1596     );
1597
1598     this->has_improvement = true;
1599     this->decoration_cleared = true;
1600     this->assets_manager_ptr->getSound("splash")->play();
1601     this->__closeBuildMenu();
1602
1603     if (not this->resource_assessed) {
1604         this->assess();
1605     }
1606
1607     this->__sendCreditsSpentMessage(build_cost);
1608     this->__sendTileStateMessage();
1609     this->__sendGameStateRequest();
1610
1611     return;
1612 }   /* __buildTidalTurbine() */
```

4.7.3.6 __buildWaveEnergyConverter()

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

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

```
1627 {
1628     int build_cost = WAVE_ENERGY_CONVERTER_BUILD_COST;
1629
1630     if (this->credits < build_cost) {
1631         std::cout « "Cannot build wave energy converter: insufficient credits (need "
1632             « build_cost « " K)" « std::endl;
1633
1634         this->__sendInsufficientCreditsMessage();
1635         return;
1636     }
1637
1638     this->tile_improvement_ptr = new WaveEnergyConverter(
1639         this->position_x,
1640         this->position_y,
1641         this->tile_resource,
1642         this->event_ptr,
```

```
1643         this->render_window_ptr,
1644         this->assets_manager_ptr,
1645         this->message_hub_ptr
1646     );
1647 
1648     this->has_improvement = true;
1649     this->decoration_cleared = true;
1650     this->assets_manager_ptr->getSound("splash")->play();
1651     this->__closeBuildMenu();
1652 
1653     if (not this->resource_assessed) {
1654         this->assess();
1655     }
1656 
1657     this->__sendCreditsSpentMessage(build_cost);
1658     this->__sendTileStateMessage();
1659     this->__sendGameStateRequest();
1660 
1661     return;
1662 }   /* __buildWaveEnergyConverter() */
```

4.7.3.7 __buildWindTurbine()

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

Helper method to build a wind turbine on this tile.

```
1514 {
1515     int build_cost = WIND_TURBINE_BUILD_COST;
1516 
1517     if (
1518         (this->tile_type == TileType :: LAKE) or
1519         (this->tile_type == TileType :: OCEAN)
1520     ) {
1521         build_cost *= WIND_TURBINE_WATER_BUILD_MULTIPLIER;
1522     }
1523 
1524     if (this->credits < build_cost) {
1525         std::cout « "Cannot build wind turbine: insufficient credits (need "
1526             « build_cost « " K)" « std::endl;
1527 
1528         this->__sendInsufficientCreditsMessage();
1529         return;
1530     }
1531 
1532     this->tile_improvement_ptr = new WindTurbine(
1533         this->position_x,
1534         this->position_y,
1535         this->tile_resource,
1536         this->event_ptr,
1537         this->render_window_ptr,
1538         this->assets_manager_ptr,
1539         this->message_hub_ptr
1540     );
1541 
1542     this->has_improvement = true;
1543     this->__closeBuildMenu();
1544 
1545     if (not this->resource_assessed) {
1546         this->assess();
1547     }
1548 
1549     if (
1550         (this->tile_type == TileType :: LAKE) or
1551         (this->tile_type == TileType :: OCEAN)
1552     ) {
1553         this->decoration_cleared = true;
1554         this->assets_manager_ptr->getSound("splash")->play();
1555     }
1556 
1557     this->__sendCreditsSpentMessage(build_cost);
1558     this->__sendTileStateMessage();
1559     this->__sendGameStateRequest();
1560 
1561     return;
1562 }   /* __buildWindTurbine() */
```

4.7.3.8 __clearDecoration()

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

Helper method to clear tile decoration.

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

4.7.3.9 __closeBuildMenu()

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

Helper method to close the tile improvement build menu.

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

4.7.3.10 __getTileCoordsSubstring()

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

Helper method to assemble and return tile coordinates substring.

Returns

Tile coordinates substring.

```
1827 {
1828     std::string coords_substring = "TILE COORDS:  (";
1829     coords_substring += std::to_string(int(this->position_x - 400));
1830     coords_substring += ", ";
1831     coords_substring += std::to_string(int(this->position_y - 400));
1832     coords_substring += ")\n";
1833 
1834     return coords_substring;
1835 }   /* __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.

```
1986 {
1987     std::string improvement_substring = "TILE IMPROVEMENT:  ";
1988 
1989     if (this->has_improvement) {
1990         improvement_substring += this->tile_improvement_ptr->tile_improvement_string;
1991         improvement_substring += "\n";
1992     }
1993 
1994     else {
1995         improvement_substring += "NONE\n";
1996     }
1997 
1998     return improvement_substring;
1999 }   /* __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.

```
2016 {
2017     //                  32 char x 17 line console "--------------------------------\n";
2018     std::string options_substring               = "     **** TILE OPTIONS ****     \n";
2019     options_substring                          += "                                \n";
2020 
2021     if (this->game_phase == "build settlement") {
2022         if (
2023             (this->tile_type != TileType :: OCEAN) and
2024             (this->tile_type != TileType :: LAKE)
2025         ) {
2026             options_substring += "[B]:  BUILD SETTLEMENT (";
2027             options_substring += std::to_string(BUILD_SETTLEMENT_COST);
2028             options_substring += " K)\n";
2029         }
```

```
2030     }
2031
2032
2033     else if (this->game_phase == "system management") {
2034         if (this->has_improvement) {
2035             options_substring.clear();
2036             options_substring = this->tile_improvement_ptr->getTileOptionsSubstring();
2037         }
2038
2039
2040         else if (not this->resource_assessed) {
2041             options_substring += "[A]:  ASSESS RESOURCE (";
2042             options_substring += std::to_string(RESOURCE_ASSESSMENT_COST);
2043             options_substring += " K)\n";
2044         }
2045
2046
2047         else if (
2048             (not this->decoration_cleared) and
2049             (this->tile_type != TileType :: OCEAN) and
2050             (this->tile_type != TileType :: LAKE)
2051         ) {
2052             options_substring += "[C]:  CLEAR TILE (";
2053
2054             switch (this->tile_type) {
2055                 case (TileType :: FOREST): {
2056                     options_substring += std::to_string(CLEAR_FOREST_COST);
2057
2058                     break;
2059                 }
2060
2061
2062                 case (TileType :: MOUNTAINS): {
2063                     options_substring += std::to_string(CLEAR_MOUNTAINS_COST);
2064
2065                     break;
2066                 }
2067
2068
2069                 case (TileType :: PLAINS): {
2070                     options_substring += std::to_string(CLEAR_PLAINS_COST);
2071
2072                     break;
2073                 }
2074
2075
2076                 default: {
2077                     //do nothing!
2078
2079                     break;
2080                 }
2081             }
2082
2083             options_substring += " K)\n";
2084         }
2085
2086
2087         else if (
2088             (this->decoration_cleared) or
2089             (this->tile_type == TileType :: OCEAN) or
2090             (this->tile_type == TileType :: LAKE)
2091         ) {
2092             options_substring += "[B]: OPEN BUILD MENU\n";
2093         }
2094     }
2095
2096
2097     else if (this->game_phase == "victory") {
2098         options_substring                       += "       **** VICTORY ****        \n";
2099     }
2100
2101
2102     else {
2103         options_substring                       += "        **** LOSS ****          \n";
2104     }
2105
2106     return options_substring;
2107 }   /* __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.

```
1916 {
1917     std::string resource_substring = "TILE RESOURCE:     ";
1918 
1919     if (this->resource_assessed) {
1920         switch (this->tile_resource) {
1921             case (TileResource :: POOR): {
1922                 resource_substring += "POOR\n";
1923 
1924                 break;
1925             }
1926 
1927 
1928             case (TileResource ::BELOW_AVERAGE): {
1929                 resource_substring += "BELOW AVERAGE\n";
1930 
1931                 break;
1932             }
1933 
1934 
1935             case (TileResource :: AVERAGE): {
1936                 resource_substring += "AVERAGE\n";
1937 
1938                 break;
1939             }
1940 
1941 
1942             case (TileResource :: ABOVE_AVERAGE): {
1943                 resource_substring += "ABOVE AVERAGE\n";
1944 
1945                 break;
1946             }
1947 
1948 
1949             case (TileResource :: GOOD): {
1950                 resource_substring += "GOOD\n";
1951 
1952                 break;
1953             }
1954 
1955 
1956             default: {
1957                 resource_substring += "???\n";
1958 
1959                 break;
1960             }
1961         }
1962     }
1963 
1964     else {
1965         resource_substring += "???\n";
1966     }
1967 
1968     return resource_substring;
1969 }   /* __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.

```
1852 {
1853     std::string type_substring = "TILE TYPE:         ";
1854 
1855     switch (this->tile_type) {
1856         case (TileType :: FOREST): {
1857             type_substring += "FOREST\n";
1858 
1859             break;
1860         }
1861 
1862 
1863         case (TileType :: LAKE): {
1864             type_substring += "LAKE\n";
1865 
1866             break;
1867         }
1868 
1869 
1870         case (TileType :: MOUNTAINS): {
1871             type_substring += "MOUNTAINS\n";
1872 
1873             break;
1874         }
1875 
1876 
1877         case (TileType :: OCEAN): {
1878             type_substring += "OCEAN\n";
1879 
1880             break;
1881         }
1882 
1883 
1884         case (TileType :: PLAINS): {
1885             type_substring += "PLAINS\n";
1886 
1887             break;
1888         }
1889 
1890 
1891         default: {
1892             type_substring += "???\n";
1893 
1894             break;
1895         }
1896     }
1897 
1898     return type_substring;
1899 }   /* __getTileTypeSubstring() */
```

4.7.3.15 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
874 {
875     if (not this->is_selected) {
876         return;
877     }
878 
879 
880     if (this->event_ptr->key.code == sf::Keyboard::Escape) {
881         this->__setIsSelected(false);
882     }
883 
884 
885     if (this->build_menu_open) {
886         switch (this->tile_type) {
887             case (TileType :: FOREST): {
888                 switch (this->event_ptr->key.code) {
889                     case (sf::Keyboard::D): {
890                         this->__buildDieselGenerator();
891 
892                         break;
```

```
893                     }
894
895
896                     case (sf::Keyboard::S): {
897                         this->__buildSolarPV();
898
899                         break;
900                     }
901
902
903                     case (sf::Keyboard::W): {
904                         this->__buildWindTurbine();
905
906                         break;
907                     }
908
909
910                     case (sf::Keyboard::E): {
911                         this->__buildEnergyStorage();
912
913                         break;
914                     }
915
916
917                     default: {
918                         // do nothing!
919
920                         break;
921                     }
922                 }
923
924                 break;
925             }
926
927
928             case (TileType :: LAKE): {
929                 switch (this->event_ptr->key.code) {
930                     case (sf::Keyboard::S): {
931                         this->__buildSolarPV();
932
933                         break;
934                     }
935
936
937                     case (sf::Keyboard::W): {
938                         this->__buildWindTurbine();
939
940                         break;
941                     }
942
943
944                     default: {
945                         // do nothing!
946
947                         break;
948                     }
949                 }
950
951                 break;
952             }
953
954
955             case (TileType :: MOUNTAINS): {
956                 switch (this->event_ptr->key.code) {
957                     case (sf::Keyboard::D): {
958                         this->__buildDieselGenerator();
959
960                         break;
961                     }
962
963
964                     case (sf::Keyboard::S): {
965                         this->__buildSolarPV();
966
967                         break;
968                     }
969
970
971                     case (sf::Keyboard::W): {
972                         this->__buildWindTurbine();
973
974                         break;
975                     }
976
977
978                     case (sf::Keyboard::E): {
979                         this->__buildEnergyStorage();
```

```
980
981                         break;
982                     }
983
984
985                     default: {
986                         // do nothing!
987
988                         break;
989                     }
990                 }
991
992                 break;
993             }
994
995
996             case (TileType :: OCEAN): {
997                 switch (this->event_ptr->key.code) {
998                     case (sf::Keyboard::W): {
999                         this->__buildWindTurbine();
1000
1001                         break;
1002                     }
1003
1004
1005                     case (sf::Keyboard::T): {
1006                         this->__buildTidalTurbine();
1007
1008                         break;
1009                     }
1010
1011
1012                     case (sf::Keyboard::A): {
1013                         this->__buildWaveEnergyConverter();
1014
1015                         break;
1016                     }
1017
1018
1019                     default: {
1020                         // do nothing!
1021
1022                         break;
1023                     }
1024                 }
1025
1026                 break;
1027             }
1028
1029
1030             case (TileType :: PLAINS): {
1031                 switch (this->event_ptr->key.code) {
1032                     case (sf::Keyboard::D): {
1033                         this->__buildDieselGenerator();
1034
1035                         break;
1036                     }
1037
1038
1039                     case (sf::Keyboard::S): {
1040                         this->__buildSolarPV();
1041
1042                         break;
1043                     }
1044
1045
1046                     case (sf::Keyboard::W): {
1047                         this->__buildWindTurbine();
1048
1049                         break;
1050                     }
1051
1052
1053                     case (sf::Keyboard::E): {
1054                         this->__buildEnergyStorage();
1055
1056                         break;
1057                     }
1058
1059
1060                     default: {
1061                         // do nothing!
1062
1063                         break;
1064                     }
1065                 }
1066
```

```
1067                 break;
1068             }
1069
1070
1071             default: {
1072                 //do nothing!
1073
1074                 break;
1075             }
1076         }
1077     }
1078
1079
1080     if (this->game_phase == "build settlement") {
1081         if (
1082             (this->tile_type != TileType :: OCEAN) and
1083             (this->tile_type != TileType :: LAKE)
1084         ) {
1085             if (this->event_ptr->key.code == sf::Keyboard::B) {
1086                 this->__buildSettlement();
1087             }
1088         }
1089     }
1090
1091
1092     else if (this->game_phase == "system management") {
1093         if (this->has_improvement) {
1094             if (this->tile_improvement_ptr->tile_improvement_type != TileImprovementType :: SETTLEMENT)
      {
1095                 if (this->event_ptr->key.code == sf::Keyboard::P) {
1096                     this->__scrapImprovement();
1097                 }
1098             }
1099
1100             /*
1101              * All other inputs will be caught and handled by
1102              *   this->tile_improvement_ptr->processEvent()
1103              */
1104         }
1105
1106
1107         else if (not this->resource_assessed) {
1108             if (this->event_ptr->key.code == sf::Keyboard::A) {
1109                 if (this->credits < RESOURCE_ASSESSMENT_COST) {
1110                     std::cout « "Cannot assess resource: insufficient credits (need "
1111                         « RESOURCE_ASSESSMENT_COST « " K)" « std::endl;
1112
1113                     this->__sendInsufficientCreditsMessage();
1114                 }
1115
1116                 else {
1117                     this->assess();
1118                     this->__sendCreditsSpentMessage(RESOURCE_ASSESSMENT_COST);
1119                     this->__sendTileStateMessage();
1120                     this->__sendGameStateRequest();
1121                 }
1122             }
1123         }
1124
1125
1126         else if (
1127             (not this->decoration_cleared) and
1128             (this->tile_type != TileType :: OCEAN) and
1129             (this->tile_type != TileType :: LAKE)
1130         ) {
1131             if (this->event_ptr->key.code == sf::Keyboard::C) {
1132                 int clear_cost = 0;
1133
1134                 switch (this->tile_type) {
1135                     case (TileType :: FOREST): {
1136                         clear_cost = CLEAR_FOREST_COST;
1137
1138                         break;
1139                     }
1140
1141
1142                     case (TileType :: MOUNTAINS): {
1143                         clear_cost = CLEAR_MOUNTAINS_COST;
1144
1145                         break;
1146                     }
1147
1148
1149                     case (TileType :: PLAINS): {
1150                         clear_cost = CLEAR_PLAINS_COST;
1151
1152                         break;
```

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

4.7.3.16 __handleKeyReleaseEvents()

```
void HexTile::__handleKeyReleaseEvents (
             void  )  [private]
1198 {
1199     if (not this->is_selected) {
1200         return;
1201     }
1202
1203
1204     switch (this->event_ptr->key.code) {
1205         case (sf::Keyboard::P): {
1206             if (this->has_improvement) {
1207                 this->scrap_improvement_frame = 0;
1208
1209                 if (
1210                     this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1211                 ) {
1212                     this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1213                         sf::Color(255, 255, 255, 255)
1214                     );
1215                 }
1216
1217                 else {
1218                     for (
1219                         size_t i = 0;
1220                         i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1221                         i++
1222                     ) {
1223                         this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1224                             sf::Color(255, 255, 255, 255)
1225                         );
1226                     }
1227                 }
1228             }
1229
1230
1231             break;
```

```
1232         }
1233
1234
1235         default: {
1236             // do nothing!
1237
1238             break;
1239         }
1240     }
1241
1242     /*
1243     if (this->event_ptr->key.code == sf::Keyboard::P) {
1244
1245     }
1246     */
1247
1248     return;
1249 }   /* __handleKeyReleaseEvents() */
```

4.7.3.17 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.7.3.18 __isClicked()

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

Helper method to determine if tile was clicked on.

Returns

Boolean indicating whether or not tile was clicked on.

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

4.7.3.19 __openBuildMenu()

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

Helper method to open the tile improvement build menu.

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

4.7.3.20 __scrapImprovement()

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

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

```
1726 {
1727     //  1. implement key hold confirmation
1728     if (this->scrap_improvement_frame <= FRAMES_PER_SECOND) {
1729         double colour_scalar =
1730             1 - ((double)(this->scrap_improvement_frame) / (FRAMES_PER_SECOND));
1731 
1732         if (
1733             this->tile_improvement_ptr->tile_improvement_sprite_static.getTexture() != NULL
1734         ) {
1735             this->tile_improvement_ptr->tile_improvement_sprite_static.setColor(
1736                 sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
1737             );
1738         }
1739 
1740         else {
1741             for (
1742                 size_t i = 0;
1743                 i < this->tile_improvement_ptr->tile_improvement_sprite_animated.size();
1744                 i++
1745             ) {
1746                 this->tile_improvement_ptr->tile_improvement_sprite_animated[i].setColor(
1747                     sf::Color(255, 255 * colour_scalar, 255 * colour_scalar, 255)
```

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

4.7.3.21 __sendAssessNeighboursMessage()

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

Helper method to format and send assess neighbours message.

```
2163 {
2164     Message assess_neighbours_message;
2165
2166     assess_neighbours_message.channel = HEX_MAP_CHANNEL;
2167     assess_neighbours_message.subject = "assess neighbours";
2168
2169     this->message_hub_ptr->sendMessage(assess_neighbours_message);
2170
2171     std::cout « "Assess neighbours message sent by " « this « std::endl;
2172
2173     return;
2174 }   /* __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.

```
2246 {
2247     Message credits_spent_message;
2248 
2249     credits_spent_message.channel = GAME_CHANNEL;
2250     credits_spent_message.subject = "credits spent";
2251 
2252     credits_spent_message.int_payload["credits spent"] = credits_spent;
2253 
2254     this->message_hub_ptr->sendMessage(credits_spent_message);
2255 
2256     std::cout << "Credits spent (" << credits_spent << ") message sent by " << this
2257         << std::endl;
2258     return;
2259 }   /* __sendCreditsSpentMessage() */
```

4.7.3.23 __sendGameStateRequest()

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

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

```
2189 {
2190     Message game_state_request;
2191 
2192     game_state_request.channel = GAME_CHANNEL;
2193     game_state_request.subject = "state request";
2194 
2195     this->message_hub_ptr->sendMessage(game_state_request);
2196 
2197     std::cout << "Game state request message sent by " << this << std::endl;
2198     return;
2199 }   /* __sendGameStateRequest() */
```

4.7.3.24 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

```
2274 {
2275     Message insufficient_credits_message;
2276 
2277     insufficient_credits_message.channel = GAME_CHANNEL;
2278     insufficient_credits_message.subject = "insufficient credits";
2279 
2280     this->message_hub_ptr->sendMessage(insufficient_credits_message);
2281 
2282     std::cout << "Insufficient credits message sent by " << this << std::endl;
2283 
2284     return;
2285 }   /* __sendInsufficientCreditsMessage() */
```

4.7.3.25 __sendTileSelectedMessage()

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

Helper method to format and send message on tile selection.

```
1801 {
1802     Message tile_selected_message;
1803 
1804     tile_selected_message.channel = TILE_SELECTED_CHANNEL;
1805     tile_selected_message.subject = "tile selected";
1806 
1807     this->message_hub_ptr->sendMessage(tile_selected_message);
1808 
1809     return;
1810 }   /* __sendTileSelectedMessage() */
```

4.7.3.26 __sendTileStateMessage()

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

Helper method to format and send tile state message.

```
2122 {
2123     Message tile_state_message;
2124 
2125     tile_state_message.channel = TILE_STATE_CHANNEL;
2126     tile_state_message.subject = "tile state";
2127 
2128 
2129     //                  32 char x 17 line console "--------------------------------\n";
2130     std::string console_string                    = "      **** TILE INFO ****       \n";
2131 
2132     console_string                               += this->__getTileCoordsSubstring();
2133     console_string                               += "                                \n";
2134 
2135     console_string                               += this->__getTileTypeSubstring();
2136     console_string                               += this->__getTileResourceSubstring();
2137     console_string                               += this->__getTileImprovementSubstring();
2138     console_string                               += "                                \n";
2139 
2140     console_string                               += this->__getTileOptionsSubstring();
2141 
2142     tile_state_message.string_payload["console string"] = console_string;
2143 
2144     this->message_hub_ptr->sendMessage(tile_state_message);
2145 
2146     std::cout « "Tile state message sent by " « this « std::endl;
2147     return;
2148 }   /* __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.

```
2216 {
2217     Message update_game_phase_message;
2218 
2219     update_game_phase_message.channel = GAME_CHANNEL;
2220     update_game_phase_message.subject = "update game phase";
2221 
2222     update_game_phase_message.string_payload["game phase"] = game_phase;
2223 
2224     this->message_hub_ptr->sendMessage(update_game_phase_message);
2225 
2226     std::cout « "Update game phase message sent by " « this « std::endl;
2227 
2228     return;
2229 }   /* __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.

```
2708 {
2709     this->resource_assessed = true;
2710     this->resource_assessment = true;
2711 
2712     this->assets_manager_ptr->getSound("resource assessment")->play();
2713 
2714     this->__setResourceText();
2715     this->__sendTileStateMessage();
2716 
2717     return;
2718 }   /* assess() */
```

4.7.3.45 decorateTile()

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

Method to decorate tile.

```
2586 {
2587     switch (this->tile_type) {
2588         case (TileType :: FOREST): {
2589             this->tile_decoration_sprite.setTexture(
2590                 *(this->assets_manager_ptr->getTexture("pine_tree_64x64_1"))
2591             );
2592 
2593             break;
2594         }
2595 
2596         case (TileType :: LAKE): {
2597             this->tile_decoration_sprite.setTexture(
2598                 *(this->assets_manager_ptr->getTexture("water_shimmer_64x64_1"))
2599             );
2600 
2601             break;
2602         }
2603 
2604         case (TileType :: MOUNTAINS): {
2605             this->tile_decoration_sprite.setTexture(
2606                 *(this->assets_manager_ptr->getTexture("mountain_64x64_1"))
2607             );
2608 
2609             break;
2610         }
2611 
2612         case (TileType :: OCEAN): {
2613             this->tile_decoration_sprite.setTexture(
2614                 *(this->assets_manager_ptr->getTexture("water_waves_64x64_1"))
2615             );
2616 
2617             break;
2618         }
2619 
2620         case (TileType :: PLAINS): {
2621             this->tile_decoration_sprite.setTexture(
2622                 *(this->assets_manager_ptr->getTexture("wheat_64x64_1"))
2623             );
2624 
2625             break;
2626         }
2627 
2628         default: {
2629             // do nothing!
2630 
2631             break;
2632         }
2633     }
2634 
2635 
2636     if (this->tile_type == TileType :: OCEAN or this->tile_type == TileType :: LAKE) {
```

```
2637         this->tile_decoration_sprite.setOrigin(
2638             this->tile_decoration_sprite.getLocalBounds().width / 2,
2639             this->tile_decoration_sprite.getLocalBounds().height / 2
2640         );
2641 
2642         this->tile_decoration_sprite.setPosition(
2643             this->position_x,
2644             this->position_y
2645         );
2646 
2647         if ((double)rand() / RAND_MAX > 0.5) {
2648             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2649         }
2650     }
2651 
2652     else {
2653         this->tile_decoration_sprite.setOrigin(
2654             this->tile_decoration_sprite.getLocalBounds().width / 2,
2655             this->tile_decoration_sprite.getLocalBounds().height
2656         );
2657 
2658         this->tile_decoration_sprite.setPosition(
2659             this->position_x,
2660             this->position_y + 12
2661         );
2662 
2663         if ((double)rand() / RAND_MAX > 0.5) {
2664             this->tile_decoration_sprite.setScale(sf::Vector2f(-1, 1));
2665         }
2666     }
2667 
2668     return;
2669 }   /* 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.

```
2849 {
2850     //  1. draw hex
2851     this->render_window_ptr->draw(this->tile_sprite);
2852 
2853     //  2. draw node
2854     if (this->show_node) {
2855         this->render_window_ptr->draw(this->node_sprite);
2856     }
2857 
2858     //  3. draw tile decoration
2859     if (not this->decoration_cleared) {
2860         this->render_window_ptr->draw(this->tile_decoration_sprite);
2861     }
2862 
2863     //  4. draw selection outline
2864     if (this->is_selected) {
2865         sf::Color outline_colour = this->select_outline_sprite.getOutlineColor();
2866 
2867         outline_colour.a =
2868             255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2);
2869 
2870         this->select_outline_sprite.setOutlineColor(outline_colour);
2871 
2872         this->render_window_ptr->draw(this->select_outline_sprite);
2873     }
2874 
2875     //  5. draw tile improvement
2876     if (this->has_improvement) {
2877         if (not this->tile_improvement_ptr->just_built) {
2878             this->tile_improvement_ptr->draw();
2879         }
2880     }
2881 
2882     //  6. draw resource
2883     if (this->show_resource) {
2884         this->render_window_ptr->draw(this->resource_chip_sprite);
2885         this->render_window_ptr->draw(this->resource_text);
2886     }
```

```
2887
2888     //  7. draw resource assessment notification
2889     if (this->resource_assessment) {
2890         int alpha = this->magnifying_glass_sprite.getColor().a;
2891
2892         alpha -= 0.05 * FRAMES_PER_SECOND;
2893         if (alpha < 0) {
2894             alpha = 0;
2895             this->resource_assessment = false;
2896         }
2897
2898         this->magnifying_glass_sprite.setColor(
2899             sf::Color(255, 255, 255, alpha)
2900         );
2901
2902         this->render_window_ptr->draw(this->magnifying_glass_sprite);
2903     }
2904
2905     //  8. draw explosion, then settlement placement
2906     if (this->draw_explosion) {
2907         this->render_window_ptr->draw(this->explosion_sprite_reel[this->explosion_frame]);
2908
2909         if (this->frame % (FRAMES_PER_SECOND / 20) == 0) {
2910             this->explosion_frame++;
2911         }
2912
2913         if (this->explosion_frame >= this->explosion_sprite_reel.size()) {
2914             this->draw_explosion = false;
2915             this->explosion_frame = 0;
2916         }
2917     }
2918
2919     else if (this->has_improvement) {
2920         if (this->tile_improvement_ptr->just_built) {
2921             this->tile_improvement_ptr->draw();
2922         }
2923     }
2924
2925     //  9. build menu
2926     if (this->build_menu_open) {
2927         this->render_window_ptr->draw(this->build_menu_backing);
2928         this->render_window_ptr->draw(this->build_menu_backing_text);
2929
2930         for (size_t i = 0; i < this->build_menu_options_vec.size(); i++) {
2931             for (size_t j = 0; j < this->build_menu_options_vec[i].size(); j++) {
2932                 this->render_window_ptr->draw(this->build_menu_options_vec[i][j]);
2933             }
2934             this->render_window_ptr->draw(this->build_menu_options_text_vec[i]);
2935         }
2936     }
2937
2938     this->frame++;
2939     return;
2940 }   /* draw() */
```

4.7.3.47 processEvent()

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

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

```
2733 {
2734     //  1. process TileImprovement events
2735     if (
2736         this->is_selected and
2737         this->tile_improvement_ptr != NULL
2738     ) {
2739         this->tile_improvement_ptr->processEvent();
2740     }
2741
2742     //  2. process HexTile events
2743     if (this->event_ptr->type == sf::Event::KeyPressed) {
2744         this->__handleKeyPressEvents();
2745     }
2746
2747     if (this->event_ptr->type == sf::Event::KeyReleased) {
2748         this->__handleKeyReleaseEvents();
2749     }
```

```
2750
2751     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
2752         this->__handleMouseButtonEvents();
2753     }
2754
2755     return;
2756 }   /* processEvent() */
```

4.7.3.48 processMessage()

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

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

```
2771 {
2772     //  1. process TileImprovement messages
2773     if (this->tile_improvement_ptr != NULL) {
2774         this->tile_improvement_ptr->processMessage();
2775     }
2776
2777     //  2. process HexTile messages
2778     if (this->is_selected) {
2779         if (not this->message_hub_ptr->isEmpty(TILE_STATE_CHANNEL)) {
2780             Message tile_state_message = this->message_hub_ptr->receiveMessage(
2781                 TILE_STATE_CHANNEL
2782             );
2783
2784             if (tile_state_message.subject == "state request") {
2785                 this->__sendTileStateMessage();
2786
2787                 std::cout « "Tile state request received by " « this « std::endl;
2788                 this->message_hub_ptr->popMessage(TILE_STATE_CHANNEL);
2789             }
2790         }
2791
2792         std::cout « "Current credits (HexTile): " « this->credits « " K" «
2793             std::endl;
2794     }
2795
2796     if (not this->message_hub_ptr->isEmpty(GAME_STATE_CHANNEL)) {
2797         Message game_state_message = this->message_hub_ptr->receiveMessage(
2798             GAME_STATE_CHANNEL
2799         );
2800
2801         if (game_state_message.subject == "game state") {
2802             this->credits = game_state_message.int_payload["credits"];
2803             this->game_phase = game_state_message.string_payload["game phase"];
2804
2805             if (this->tile_improvement_ptr != NULL) {
2806                 this->tile_improvement_ptr->credits = this->credits;
2807                 this->tile_improvement_ptr->game_phase = this->game_phase;
2808
2809                 this->tile_improvement_ptr->month =
2810                     game_state_message.int_payload["month"];
2811
2812                 this->tile_improvement_ptr->demand_MWh =
2813                     game_state_message.int_payload["demand_MWh"];
2814
2815                 this->tile_improvement_ptr->demand_vec_MWh =
2816                     game_state_message.vector_payload["demand_vec_MWh"];
2817
2818                 this->tile_improvement_ptr->update();
2819             }
2820
2821             this->message_hub_ptr->incrementMessageRead(GAME_STATE_CHANNEL);
2822
2823             std::cout « "Game state message read and passed by " « this «
2824                 " (credits: " « this->credits « " K)" « std::endl;
2825
2826             if (this->is_selected) {
2827                 this->__sendTileStateMessage();
2828             }
2829         }
2830     }
2831
2832     return;
2833 }   /* 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].

```
2535 {
2536     //  1. check input
2537     if (input_value < 0 or input_value > 1) {
2538         std::string error_str = "ERROR  HexTile::setTileResource()  given input value is ";
2539         error_str += "not in the closed interval [0, 1]";
2540 
2541         #ifdef _WIN32
2542             std::cout « error_str « std::endl;
2543         #endif  /* _WIN32 */
2544 
2545         throw std::runtime_error(error_str);
2546     }
2547 
2548     //  2. convert input value to tile resource
2549     TileResource tile_resource;
2550 
2551     if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[0]) {
2552         tile_resource = TileResource :: POOR;
2553     }
2554     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[1]) {
2555         tile_resource = TileResource :: BELOW_AVERAGE;
2556     }
2557     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[2]) {
2558         tile_resource = TileResource :: AVERAGE;
2559     }
2560     else if (input_value <= TILE_RESOURCE_CUMULATIVE_PROBABILITIES[3]) {
2561         tile_resource = TileResource :: ABOVE_AVERAGE;
2562     }
2563     else {
2564         tile_resource = TileResource :: GOOD;
2565     }
2566 
2567     //  3. call alternate method
2568     this->setTileResource(tile_resource);
2569 
2570     return;
2571 }   /* 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.

```
2513 {
2514     this->tile_resource = tile_resource;
2515     this->__setResourceText();
2516 
2517     return;
2518 }   /* 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].

```
2463 {
2464     //  1. check input
2465     if (input_value < 0 or input_value > 1) {
2466         std::string error_str = "ERROR  HexTile::setTileType()  given input value is ";
2467         error_str += "not in the closed interval [0, 1]";
2468 
2469         #ifdef _WIN32
2470             std::cout « error_str « std::endl;
2471         #endif  /* _WIN32 */
2472 
2473         throw std::runtime_error(error_str);
2474     }
2475 
2476     //  2. convert input value to tile type
2477     TileType tile_type;
2478 
2479     if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[0]) {
2480         tile_type = TileType :: LAKE;
2481     }
2482     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[1]) {
2483         tile_type = TileType :: PLAINS;
2484     }
2485     else if (input_value <= TILE_TYPE_CUMULATIVE_PROBABILITIES[2]) {
2486         tile_type = TileType :: FOREST;
2487     }
2488     else {
2489         tile_type = TileType :: MOUNTAINS;
2490     }
2491 
2492     //  3. call alternate method
2493     this->setTileType(tile_type);
2494 
2495     return;
2496 }   /* 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.

```
2402 {
2403     this->tile_type = tile_type;
2404 
2405     switch (this->tile_type) {
2406         case (TileType :: FOREST): {
2407             this->tile_sprite.setFillColor(FOREST_GREEN);
2408 
2409             break;
2410         }
2411 
2412         case (TileType :: LAKE): {
```

```
2413             this->tile_sprite.setFillColor(LAKE_BLUE);
2414
2415             break;
2416         }
2417
2418         case (TileType :: MOUNTAINS): {
2419             this->tile_sprite.setFillColor(MOUNTAINS_GREY);
2420
2421             break;
2422         }
2423
2424         case (TileType :: OCEAN): {
2425             this->tile_sprite.setFillColor(OCEAN_BLUE);
2426
2427             break;
2428         }
2429
2430         case (TileType :: PLAINS): {
2431             this->tile_sprite.setFillColor(PLAINS_YELLOW);
2432
2433             break;
2434         }
2435
2436         default: {
2437             // do nothing!
2438
2439             break;
2440         }
2441     }
2442
2443     this->__setUpBuildMenu();
2444
2445     return;
2446 }   /* setTileType(TileType) */
```

4.7.3.53 toggleResourceOverlay()

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

Method to toggle the tile resource overlay.

```
2684 {
2685     if (this->show_resource) {
2686         this->show_resource = false;
2687     }
2688     else {
2689         this->show_resource = true;
2690     }
2691
2692     return;
2693 }   /* toggleResourceOverlay() */
```

4.7.4 Member Data Documentation

4.7.4.1 assets_manager_ptr

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

A pointer to the assets manager.

4.7.4.2 build_menu_backing

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

A backing for the tile build menu.

4.7.4.3 build_menu_backing_text

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

A text label for the build menu.

4.7.4.4 build_menu_open

```
bool HexTile::build_menu_open
```

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

4.7.4.5 build_menu_options_text_vec

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

A vector of text for the tile build options.

4.7.4.6 build_menu_options_vec

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

A vector of sprites for illustrating the tile build options.

4.7.4.7 credits

```
int HexTile::credits
```

The current balance of credits.

4.7.4.8 decoration_cleared

```
bool HexTile::decoration_cleared
```

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

4.7.4.9 draw_explosion

```
bool HexTile::draw_explosion
```

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

4.7.4.10 event_ptr

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

A pointer to the event class.

4.7.4.11 explosion_frame

```
size_t HexTile::explosion_frame
```

The current frame of the explosion animation.

4.7.4.12 explosion_sprite_reel

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

A reel of sprites for a tile explosion animation.

4.7.4.13 frame

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

The current frame of this object.

4.7.4.14 game_phase

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

The current phase of the game.

4.7.4.15 has_improvement

```
bool HexTile::has_improvement
```

A boolean which indicates if tile has improvement or not.

4.7.4.16 is_selected

```
bool HexTile::is_selected
```

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

4.7.4.17 magnifying_glass_sprite

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

A magnifying glass sprite.

4.7.4.18 major_radius

```
double HexTile::major_radius
```

The radius of the smallest bounding circle.

4.7.4.19 message_hub_ptr

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

A pointer to the message hub.

4.7.4.20 minor_radius

```
double HexTile::minor_radius
```

The radius of the largest inscribed circle.

4.7.4.21 node_sprite

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

A circle shape to mark the tile node.

4.7.4.22 position_x

```
double HexTile::position_x
```

The x position of the tile.

4.7.4.23 position_y

```
double HexTile::position_y
```

The y position of the tile.

4.7.4.24 render_window_ptr

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

A pointer to the render window.

4.7.4.25 resource_assessed

```
bool HexTile::resource_assessed
```

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

4.7.4.26 resource_assessment

```
bool HexTile::resource_assessment
```

A boolean which triggers a resource assessment notification.

4.7.4.27 resource_chip_sprite

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

A circle shape which represents a resource chip.

4.7.4.28 resource_text

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

A text representation of the resource.

4.7.4.29 scrap_improvement_frame

```
int HexTile::scrap_improvement_frame
```

A frame for key-hold to confirm scrapping.

4.7.4.30 select_outline_sprite

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

A convex shape which outlines the tile when selected.

4.7.4.31 show_node

```
bool HexTile::show_node
```

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

4.7.4.32 show_resource

```
bool HexTile::show_resource
```

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

4.7.4.33 tile_decoration_sprite

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

A tile decoration sprite.

4.7.4.34 tile_improvement_ptr

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

A pointer to the improvement for this tile.

4.7.4.35 tile_resource

```
TileResource HexTile::tile_resource
```

The renewable resource quality of the tile.

4.7.4.36 tile_sprite

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

A convex shape which represents the tile.

4.7.4.37 tile_type

```
TileType HexTile::tile_type
```

The terrain type of the tile.

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

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

4.8 Message Struct Reference

A structure which defines a standard message format.

`#include <MessageHub.h>`

Public Attributes

- std::string channel = ""
  A string identifying the appropriate channel for this message.
- std::string subject = ""
  A string describing the message subject.
- unsigned int number_of_reads = 0
  The number of times the message has been read.
- std::map< std::string, bool > bool_payload = {}
  A boolean payload.
- std::map< std::string, int > int_payload = {}
  An int payload.
- std::map< std::string, double > double_payload = {}
  A double payload.
- std::map< std::string, std::vector< double > > vector_payload = {}
  A vector (double) payload.
- std::map< std::string, std::string > string_payload = {}
  A string payload.

4.8.1 Detailed Description

A structure which defines a standard message format.

4.8.2 Member Data Documentation

4.8.2.1 bool_payload

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

A boolean payload.

4.8.2.2 channel

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

A string identifying the appropriate channel for this message.

4.8.2.3 double_payload

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

A double payload.

4.8.2.4 int_payload

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

An int payload.

4.8.2.5 number_of_reads

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

The number of times the message has been read.

4.8.2.6 string_payload

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

A string payload.

4.8.2.7 subject

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

A string describing the message subject.

4.8.2.8 vector_payload

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

A vector (double) payload.

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

- header/ESC_core/MessageHub.h

4.9 MessageHub Class Reference

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

`#include <MessageHub.h>`

Public Member Functions

- MessageHub (void)

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

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

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

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

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

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

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

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

  Method to increment the number of times the first message in the channel has been read. Channels are implemented in a first in, first out manner (i.e. message queue).
- void popMessage (std::string)

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

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

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

  Destructor for the MessageHub class.

Private Attributes

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

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

4.9.1 Detailed Description

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

4.9.2 Constructor & Destructor Documentation

4.9.2.1 MessageHub()

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

Constructor for the MessageHub class.

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

4.9.2.2 ~MessageHub()

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

Destructor for the MessageHub class.

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

4.9.3 Member Function Documentation

4.9.3.1 addChannel()

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

Method to add channel to message map.

Parameters

channel	The key for the message channel being added.

```
129 {
130     //  1. check if channel is in map (if so, throw error)
131     if (this->message_map.count(channel) > 0) {
132         std::string error_str = "ERROR  MessageHub::addChannel()  channel ";
133         error_str += channel;
134         error_str += " is already in message map";
135
```

```
136         #ifdef _WIN32
137             std::cout « error_str « std::endl;
138         #endif  /* _WIN32 */
139 
140         throw std::runtime_error(error_str);
141     }
142 
143     //  2. add channel to map
144     this->message_map[channel] = {};
145 
146     std::cout « "Channel " « channel « " added to message hub" « std::endl;
147 
148     return;
149 }   /* addChannel() */
```

4.9.3.2 clear()

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

Method to clear the MessageHub.

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

4.9.3.3 clearMessages()

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

Method to clear messages from the MessageHub.

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

4.9.3.4 hasTraffic()

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

Method to determine if there remains any message traffic.

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

4.9.3.5 incrementMessageRead()

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

Method to increment the number of times the first message in the channel has been read. Channels are implemented in a first in, first out manner (i.e. message queue).

Parameters

channel	The key for the message channel being received from.

```
385 {
386     //  1. check if channel is in map (if not, throw error)
387     if (this->message_map.count(channel) <= 0) {
388         std::string error_str = "ERROR  MessageHub::incrementMessageRead()  channel ";
389         error_str += channel;
390         error_str += " is not in message map";
391 
392         #ifdef _WIN32
393             std::cout << error_str << std::endl;
394         #endif  /* _WIN32 */
395 
396         throw std::runtime_error(error_str);
397     }
398 
399     //  2. check if channel is empty (if so, throw error)
400     if (this->message_map[channel].empty()) {
401         std::string error_str = "ERROR  MessageHub::incrementMessageRead()  channel ";
402         error_str += channel;
403         error_str += " is empty";
404 
405         #ifdef _WIN32
406             std::cout << error_str << std::endl;
407         #endif  /* _WIN32 */
408 
409         throw std::runtime_error(error_str);
410     }
411 
412     //  3. increment number of reads
413     this->message_map[channel].front().number_of_reads++;
414 
415     return;
416 }   /* incrementMessageRead( */
```

4.9.3.6 isEmpty()

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

Method to check if channel is empty.

Parameters

channel	The key for the message channel being checked.

Returns

A boolean indicating whether the channel is empty or not.

```
295 {
296     //  1. check if channel is in map (if not, throw error)
297     if (this->message_map.count(channel) <= 0) {
298         std::string error_str = "ERROR  MessageHub::isEmpty()  channel ";
```

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

4.9.3.7 popMessage()

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

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

Parameters

channel	The key for the message channel being popped.

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

4.9.3.8 printState()

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

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

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

4.9.3.9 receiveMessage()

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

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

Parameters

channel	The key for the message channel being received from.

Returns

The first message in the given channel.

```
335 {
336     //  1. check if channel is in map (if not, throw error)
337     if (this->message_map.count(channel) <= 0) {
338         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
339         error_str += channel;
340         error_str += " is not in message map";
341
342         #ifdef _WIN32
343             std::cout « error_str « std::endl;
344         #endif  /* _WIN32 */
345
346         throw std::runtime_error(error_str);
347     }
348
349     //  2. check if channel is empty (if so, throw error)
350     if (this->message_map[channel].empty()) {
351         std::string error_str = "ERROR  MessageHub::receiveMessage()  channel ";
```

```
352         error_str += channel;
353         error_str += " is empty";
354 
355         #ifdef _WIN32
356             std::cout « error_str « std::endl;
357         #endif  /* _WIN32 */
358 
359         throw std::runtime_error(error_str);
360     }
361 
362     //  3. receive message
363     Message message = this->message_map[channel].front();
364 
365     return message;
366 }   /* receiveMessage() */
```

4.9.3.10 removeChannel()

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

Method to remove channel from message map.

Parameters

channel	The key for the message channel being removed.

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

4.9.3.11 sendMessage()

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

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

Parameters

message	The message to be sent.

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

4.9.4 Member Data Documentation

4.9.4.1 message_map

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

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

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

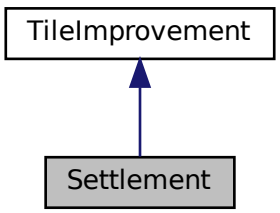
- header/ESC_core/MessageHub.h
- source/ESC_core/MessageHub.cpp

4.10 Settlement Class Reference

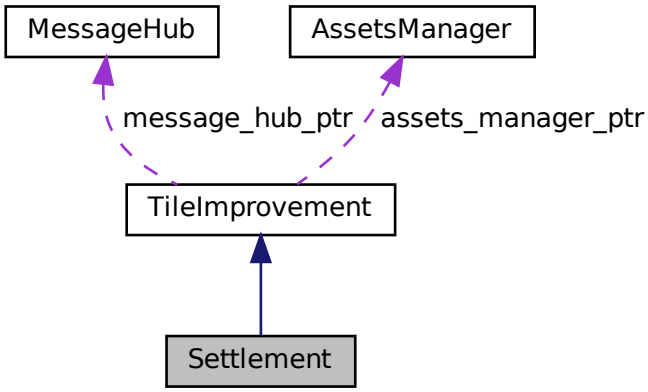
A settlement class (child class of TileImprovement).

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

Inheritance diagram for Settlement:



Collaboration diagram for Settlement:



Public Member Functions

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

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

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

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

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

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

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

  Destructor for the Settlement class.

Public Attributes

- bool draw_coin

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

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

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

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

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

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

  A coin sprite (for credits earned animation).

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

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

  Helper method to handle mouse button events.

Additional Inherited Members

4.10.1 Detailed Description

A settlement class (child class of TileImprovement).

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Settlement()

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

Constructor for the Settlement class.

Ref: Wikipedia [2023]

Parameters

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

```
241   :
242 TileImprovement(
```

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

4.10.2.2 ~Settlement()

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

Destructor for the Settlement class.

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

4.10.3 Member Function Documentation

4.10.3.1 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
131 {
132     if (this->just_built) {
133         return;
134     }
135 
136     switch (this->event_ptr->key.code) {
137         //...
```

```
138
139
140         default: {
141             // do nothing!
142
143             break;
144         }
145     }
146
147     return;
148 }   /* __handleKeyPressEvents() */
```

4.10.3.2 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.10.3.3 __setUpCoinSprite()

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

Helper method to set up and place coin sprite.

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

4.10.3.4 __setUpTileImprovementSpriteStatic()

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

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

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

4.10.3.5 draw()

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

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

Reimplemented from TileImprovement.

```
409 {
410     //  1. if just built, call base method and return
411     if (this->just_built) {
412         TileImprovement :: draw();
413 
414         return;
415     }
416 
417     //  2. draw static sprite and chimney smoke effects
418     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
419 
420     std::list<sf::Sprite>::iterator iter = this->smoke_sprite_list.begin();
421 
422     double alpha = 255;
423 
424     while (iter != this->smoke_sprite_list.end()) {
425         this->render_window_ptr->draw(*iter);
426 
427         alpha = (*iter).getColor().a;
428 
429         alpha -= this->smoke_da;
430 
431         if (alpha <= 0) {
432             iter = this->smoke_sprite_list.erase(iter);
433             continue;
434         }
435 
436         (*iter).setColor(sf::Color(255, 255, 255, alpha));
437 
438         (*iter).move(
439             this->smoke_dx + 2 * (((double)rand() / RAND_MAX) - 1) / FRAMES_PER_SECOND,
440             this->smoke_dy
441         );
442 
443         (*iter).rotate(((double)rand() / RAND_MAX));
444 
445         iter++;
446     }
```

```
447
448
449     if ((double)rand() / RAND_MAX < smoke_prob) {
450         this->smoke_sprite_list.push_back(
451             sf::Sprite(*(this->assets_manager_ptr->getTexture("emissions")))
452         );
453
454         this->smoke_sprite_list.back().setOrigin(
455             this->smoke_sprite_list.back().getLocalBounds().width / 2,
456             this->smoke_sprite_list.back().getLocalBounds().height / 2
457         );
458
459         this->smoke_sprite_list.back().setPosition(
460             this->position_x + 9 + 4 * ((double)rand() / RAND_MAX) - 2,
461             this->position_y - 33
462         );
463     }
464
465
466
467     //  4. draw coin
468     if (this->draw_coin) {
469         double alpha = this->coin_sprite.getColor().a;
470
471         alpha -= this->smoke_da;
472
473         if (alpha <= 0) {
474             this->coin_sprite.setColor(sf::Color(255, 255, 255, 255));
475             this->coin_sprite.setPosition(this->position_x, this->position_y);
476             this->draw_coin = false;
477         }
478
479         this->coin_sprite.move(0, this->smoke_dy);
480         this->coin_sprite.setColor(sf::Color(255, 255, 255, alpha));
481
482         this->render_window_ptr->draw(this->coin_sprite);
483     }
484
485     this->frame++;
486     return;
487 }   /* draw() */
```

4.10.3.6 getTileOptionsSubstring()

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

Helper method to assemble and return tile options substring.

Returns

Tile options substring.

Reimplemented from TileImprovement.

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

4.10.3.7 processEvent()

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

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

Reimplemented from TileImprovement.

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

4.10.3.8 processMessage()

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

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

Reimplemented from TileImprovement.

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

4.10.3.9 setIsSelected()

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

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

Reimplemented from TileImprovement.

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

4.10.4 Member Data Documentation

4.10.4.1 coin_sprite

`sf::Sprite Settlement::coin_sprite`

A coin sprite (for credits earned animation).

4.10.4.2 draw_coin

`bool Settlement::draw_coin`

Boolean indicating whether or not to draw credits earned coin.

4.10.4.3 smoke_da

`double Settlement::smoke_da`

The per frame delta in smoke particle alpha value.

4.10.4.4 smoke_dx

`double Settlement::smoke_dx`

The per frame delta in smoke particle x position.

4.10.4.5 smoke_dy

`double Settlement::smoke_dy`

The per frame delta in smoke particle y position.

4.10.4.6 smoke_prob

```
double Settlement::smoke_prob
```

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

4.10.4.7 smoke_sprite_list

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

A list of smoke sprite (for chimney animation).

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

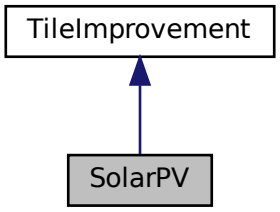
- header/Settlement.h
- source/Settlement.cpp

4.11 SolarPV Class Reference

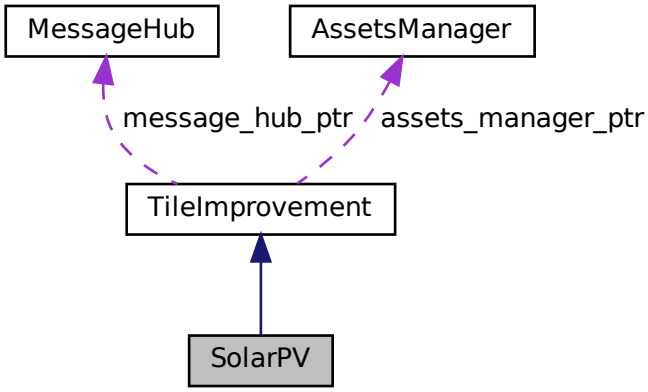
A settlement class (child class of TileImprovement).

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

Inheritance diagram for SolarPV:



Collaboration diagram for SolarPV:



Public Member Functions

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

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

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

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

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

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

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

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

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

  Destructor for the SolarPV class.

Public Attributes

- int capacity_kW

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

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

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

- int dispatchable_MWh

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

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

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

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

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

Private Member Functions

- void __setUpTileImprovementSpriteStatic (void)

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

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

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

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

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

  Helper method to repair the solar PV array.
- void __computeCapacityFactors (void)

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

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

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

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

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

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

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.11.1 Detailed Description

A settlement class (child class of TileImprovement).

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SolarPV()

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

Constructor for the SolarPV class.

Ref: Wikipedia [2023]

Parameters

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

```
759   :
760 TileImprovement(
761     position_x,
762     position_y,
763     tile_resource,
764     event_ptr,
765     render_window_ptr,
766     assets_manager_ptr,
767     message_hub_ptr
768 )
769 {
770     //  1. set attributes
771
772     //  1.1. private
773     //...
774
775     //  1.2. public
776     this->tile_improvement_type = TileImprovementType :: SOLAR_PV;
777
778     this->is_running = false;
779
780     this->health = 100;
781
782     this->capacity_kW = 100;
783     this->upgrade_level = 1;
784
785     this->storage_kWh = 0;
786     this->storage_level = 0;
787
788     this->production_MWh = 0;
789     this->dispatch_MWh = 0;
790     this->dispatchable_MWh = 0;
791
792     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
793
794     this->capacity_factor_vec.resize(30, 0);
795     this->production_vec_MWh.resize(30, 0);
```

```
796     this->dispatch_vec_MWh.resize(30, 0);
797
798     this->tile_improvement_string = "SOLAR PV ARRAY";
799
800     this->__setUpTileImprovementSpriteStatic();
801     this->__computeCapacityFactors();
802     this->update();
803
804     std::cout « "SolarPV constructed at " « this « std::endl;
805
806     return;
807 }   /* SolarPV() */
```

4.11.2.2 ~SolarPV()

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

Destructor for the SolarPV class.

```
1147 {
1148     std::cout « "SolarPV at " « this « " destroyed" « std::endl;
1149
1150     return;
1151 }   /* ~SolarPV() */
```

4.11.3 Member Function Documentation

4.11.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

```
233 {
234     TileImprovement :: __breakdown();
235
236     this->production_MWh = 0;
237     this->dispatch_MWh = 0;
238     this->dispatchable_MWh = 0;
239     this->operation_maintenance_cost = 0;
240
241     return;
242 }   /* __breakdown() */
```

4.11.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
290 {
291     if (this->is_broken) {
292         return;
293     }
294 
295     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
296     std::default_random_engine generator(seed);
297 
298     double mean =
299         this->tile_resource_scalar * MEAN_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
300 
301     double stdev = STDEV_DAILY_SOLAR_CAPACITY_FACTORS[this->month - 1];
302 
303     if (this->tile_resource_scalar > 1) {
304         stdev /= this->tile_resource_scalar;
305     }
306 
307     std::normal_distribution<double> normal_dist(mean, stdev);
308 
309     double capacity_factor = 0;
310 
311     for (int i = 0; i < 30; i++) {
312         capacity_factor = normal_dist(generator);
313 
314         if (capacity_factor < 0) {
315             capacity_factor = 0;
316         }
317 
318         this->capacity_factor_vec[i] = capacity_factor;
319     }
320 
321     return;
322 }   /* __computeCapacityFactors() */
```

4.11.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
370 {
371     if (this->is_broken) {
372         this->dispatchable_MWh = 0;
373         return;
374     }
375 
376     double stored_energy_MWh = 0;
377     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
378 
379     double demand_MWh = 0;
380     double production_MWh = 0;
381     double dispatchable_MWh = 0;
382     double difference_MWh = 0;
383 
384     double room_MWh = 0;
385 
386     for (int i = 0; i < 30; i++) {
387         demand_MWh = this->demand_vec_MWh[i];
388         production_MWh = this->production_vec_MWh[i];
389 
390         if (production_MWh <= demand_MWh) {
391             this->dispatch_vec_MWh[i] = production_MWh;
392             dispatchable_MWh += this->dispatch_vec_MWh[i];
393 
394             difference_MWh = demand_MWh - production_MWh;
395 
396             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
397                 if (difference_MWh > stored_energy_MWh) {
398                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
```

```
399                     dispatchable_MWh += stored_energy_MWh;
400                     stored_energy_MWh = 0;
401                 }
402 
403                 else {
404                     this->dispatch_vec_MWh[i] += difference_MWh;
405                     dispatchable_MWh += difference_MWh;
406                     stored_energy_MWh -= difference_MWh;
407                 }
408             }
409         }
410 
411         else {
412             this->dispatch_vec_MWh[i] = demand_MWh;
413             dispatchable_MWh += this->dispatch_vec_MWh[i];
414 
415             difference_MWh = production_MWh - demand_MWh;
416 
417             if (
418                 (storage_capacity_MWh > 0) and
419                 (stored_energy_MWh < storage_capacity_MWh)
420             ) {
421                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
422 
423                 if (difference_MWh > room_MWh) {
424                     stored_energy_MWh += room_MWh;
425                 }
426 
427                 else {
428                     stored_energy_MWh += difference_MWh;
429                 }
430             }
431         }
432     }
433 
434     this->dispatchable_MWh = round(dispatchable_MWh);
435 
436     if (this->dispatch_MWh != this->dispatchable_MWh) {
437         this->dispatch_MWh = this->dispatchable_MWh;
438     }
439 
440     return;
441 }   /* __computeDispatch() */
```

4.11.3.4 __computeProduction()

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

Helper method to compute production values.

```
337 {
338     if (this->is_broken) {
339         this->production_MWh = 0;
340         return;
341     }
342 
343     double production_MWh = 0;
344 
345     for (int i = 0; i < 30; i++) {
346         this->production_vec_MWh[i] =
347             this->max_daily_production_MWh * this->capacity_factor_vec[i];
348 
349         production_MWh += this->production_vec_MWh[i];
350     }
351 
352     this->production_MWh = round(production_MWh);
353 
354     return;
355 }   /* __computeProduction() */
```

4.11.3.5 __computeProductionCosts()

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

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

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

4.11.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

4.11.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
582 {
583     //  1. draw power capacity upgrade sprite
584     sf::Vector2f initial_position = this->tile_improvement_sprite_static.getPosition();
585     this->tile_improvement_sprite_static.setPosition(400 - 100, 400 - 32);
586 
587     sf::Color initial_colour = this->tile_improvement_sprite_static.getColor();
588     this->tile_improvement_sprite_static.setColor(sf::Color(255, 255, 255, 255));
589 
590     sf::Vector2f initial_scale = this->tile_improvement_sprite_static.getScale();
591     this->tile_improvement_sprite_static.setScale(sf::Vector2f(1, 1));
592 
593     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
594 
595     this->tile_improvement_sprite_static.setPosition(initial_position);
596     this->tile_improvement_sprite_static.setColor(initial_colour);
597     this->tile_improvement_sprite_static.setScale(initial_scale);
598 
599     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
600 
601 
602     //  2. draw power capacity upgrade text
603     //                  16 char line = "                \n"
604     std::string power_upgrade_string = "POWER CAPACITY  \n";
605     power_upgrade_string            += "                \n";
606 
607     power_upgrade_string            += "CAPACITY:  ";
608     power_upgrade_string            += std::to_string(this->capacity_kW);
609     power_upgrade_string            += " kW\n";
610 
611     power_upgrade_string            += "LEVEL:     ";
612     power_upgrade_string            += std::to_string(this->upgrade_level);
613     power_upgrade_string            += "\n\n";
614 
615     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
616         power_upgrade_string        += "[W]: + 100 kW (";
617         power_upgrade_string        +=  std::to_string(SOLAR_PV_BUILD_COST);
618         power_upgrade_string        += " K)\n";
619     }
620 
621     else {
622         power_upgrade_string        += " * MAX LEVEL *  \n";
623     }
624 
625     sf::Text power_upgrade_text = sf::Text(
626         power_upgrade_string,
627         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
628         16
629     );
630 
631     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
632     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
633     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
634 
635     this->render_window_ptr->draw(power_upgrade_text);
636 
637 
638     //  3. draw energy capacity (storage) upgrade sprite
639     this->render_window_ptr->draw(this->storage_upgrade_sprite);
640     this->render_window_ptr->draw(this->upgrade_plus_sprite);
641 
642 
643     //  4. draw energy capacity (storage) upgrade text
644     //                    16 char line = "                \n"
645     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
646     energy_upgrade_string            += "                \n";
647 
648     energy_upgrade_string            += "CAPACITY:  ";
649     energy_upgrade_string            += std::to_string(this->storage_level * 200);
650     energy_upgrade_string            += " kWh\n";
651 
652     energy_upgrade_string            += "LEVEL:     ";
653     energy_upgrade_string            += std::to_string(this->storage_level);
654     energy_upgrade_string            += "\n\n";
655 
656     if (this->storage_level < MAX_STORAGE_LEVELS) {
657         energy_upgrade_string        += "[D]: + 200 kWh (";
658         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
659         energy_upgrade_string        += " K)\n";
```

```
660     }
661
662     else {
663         energy_upgrade_string += " * MAX LEVEL *  \n";
664     }
665
666     sf::Text energy_upgrade_text = sf::Text(
667         energy_upgrade_string,
668         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
669         16
670     );
671
672     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
673     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
674     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
675
676     this->render_window_ptr->draw(energy_upgrade_text);
677
678     return;
679 }   /* __drawUpgradeOptions() */
```

4.11.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
456 {
457     if (this->just_built) {
458         return;
459     }
460
461     switch (this->event_ptr->key.code) {
462         case (sf::Keyboard::U): {
463             this->__openUpgradeMenu();
464
465             break;
466         }
467
468
469         case (sf::Keyboard::W): {
470             if (this->production_menu_open) {
471                 this->dispatch_MWh++;
472
473                 if (this->dispatch_MWh > this->dispatchable_MWh) {
474                     this->dispatch_MWh = 0;
475                 }
476
477                 this->__computeProductionCosts();
478                 this->assets_manager_ptr->getSound("interface click")->play();
479             }
480
481             else if (this->upgrade_menu_open) {
482                 this->__upgradePowerCapacity();
483             }
484
485             break;
486         }
487
488
489         case (sf::Keyboard::S): {
490             if (this->production_menu_open) {
491                 this->dispatch_MWh--;
492
493                 if (this->dispatch_MWh < 0) {
494                     this->dispatch_MWh = this->dispatchable_MWh;
495                 }
496
497                 this->__computeProductionCosts();
498                 this->assets_manager_ptr->getSound("interface click")->play();
499             }
500
501             break;
502         }
503
504
505         case (sf::Keyboard::D): {
506             if (this->upgrade_menu_open) {
```

```
507                 this->__upgradeStorageCapacity();
508                 this->__computeProduction();
509                 this->__computeDispatch();
510             }
511
512             break;
513         }
514
515
516         default: {
517             // do nothing!
518
519             break;
520         }
521     }
522
523     return;
524 }   /* __handleKeyPressEvents() */
```

4.11.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
539 {
540     if (this->just_built) {
541         return;
542     }
543
544     switch (this->event_ptr->mouseButton.button) {
545         case (sf::Mouse::Left): {
546             //...
547
548             break;
549         }
550
551
552         case (sf::Mouse::Right): {
553             //...
554
555             break;
556         }
557
558
559         default: {
560             // do nothing!
561
562             break;
563         }
564     }
565
566     return;
567 }   /* __handleMouseButtonEvents() */
```

4.11.3.10 __repair()

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

Helper method to repair the solar PV array.

Reimplemented from TileImprovement.

```
257 {
258     if (this->credits < SOLAR_PV_BUILD_COST) {
259         std::cout « "Cannot repair solar PV: insufficient credits (need "
260             « SOLAR_PV_BUILD_COST « " K)" « std::endl;
261
```

```
262         this->__sendInsufficientCreditsMessage();
263         return;
264     }
265
266     TileImprovement :: __repair();
267
268     this->just_upgraded = true;
269
270     this->__sendCreditsSpentMessage(SOLAR_PV_BUILD_COST);
271     this->__sendTileStateRequest();
272     this->__sendGameStateRequest();
273
274     return;
275 }   /* __repair() */
```

4.11.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
694 {
695     Message improvement_state_message;
696
697     improvement_state_message.channel = GAME_CHANNEL;
698     improvement_state_message.subject = "improvement state";
699
700     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
701     improvement_state_message.int_payload["operation_maintenance_cost"] =
702         this->operation_maintenance_cost;
703
704     this->message_hub_ptr->sendMessage(improvement_state_message);
705
706     std::cout « "Improvement state message sent by " « this « std::endl;
707
708     return;
709 }   /* __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.

```
912 {
913     //  1. send improvement state message
914     this->__sendImprovementStateMessage();
915 
916     //  2. update
917     this->__computeCapacityFactors();
918     this->update();
919 
920     //  3. handle start/stop
921     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
922         this->is_running = true;
923     }
924 
925     else if (this->is_running and (this->dispatch_MWh <= 0)) {
926         this->is_running = false;
927     }
928 
929     //  4. handle equipment health and breakdowns
930     if (this->is_running) {
931         this->health--;
932 
933         if (this->health <= 50) {
934             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
935 
936             if ((double)rand() / RAND_MAX <= breakdown_prob) {
937                 this->health = 0;
```

```
938             }
939         }
940 
941         if (this->health <= 0) {
942             this->__breakdown();
943         }
944     }
945 
946     //  5. send tile state request (if selected)
947     if (this->is_selected) {
948         this->__sendTileStateRequest();
949     }
950 
951     return;
952 }   /* 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.

```
1041 {
1042     //  1. if just built, call base method and return
1043     if (this->just_built) {
1044         TileImprovement :: draw();
1045 
1046         return;
1047     }
1048 
1049 
1050     //  2. handle upgrade effects
1051     if (this->just_upgraded) {
1052         this->tile_improvement_sprite_static.setColor(
1053             sf::Color(
1054                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1055                 255,
1056                 255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1057                 255
1058             )
1059         );
1060 
1061         this->tile_improvement_sprite_static.setScale(
1062             sf::Vector2f(
1063                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1064                 1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1065             )
1066         );
1067 
1068         this->upgrade_frame++;
1069     }
1070 
1071     if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1072         this->tile_improvement_sprite_static.setColor(
1073             sf::Color(255,255,255,255)
1074         );
1075 
1076         this->tile_improvement_sprite_static.setScale(sf::Vector2f(1,1));
1077 
1078         this->just_upgraded = false;
1079         this->upgrade_frame = 0;
1080     }
1081 
1082 
1083     //  3. draw static sprite
1084     this->render_window_ptr->draw(this->tile_improvement_sprite_static);
1085 
1086 
1087     //  4. draw storage upgrades
1088     for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1089         this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1090     }
1091 
1092 
1093     //  5. handle dispatch illustration
```

```
1094     if (this->dispatch_MWh > 0) {
1095         this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1096         this->__drawDispatch();
1097     }
1098 
1099 
1100     //  6. draw production menu
1101     if (this->production_menu_open) {
1102         this->render_window_ptr->draw(this->production_menu_backing);
1103         this->render_window_ptr->draw(this->production_menu_backing_text);
1104 
1105         this->__drawProductionMenu();
1106     }
1107 
1108 
1109     //  7. draw upgrade menu
1110     if (this->upgrade_menu_open) {
1111         this->render_window_ptr->draw(this->upgrade_menu_backing);
1112         this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1113 
1114         this->__drawUpgradeOptions();
1115     }
1116 
1117 
1118     //  10. handle broken effects
1119     if (this->is_broken) {
1120         this->tile_improvement_sprite_static.setColor(
1121             sf::Color(
1122                 255,
1123                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1124                 255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1125                 255
1126             )
1127         );
1128     }
1129 
1130     this->frame++;
1131     return;
1132 }   /* 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.

```
824 {
825     //                  32 char x 17 line console "--------------------------------\n";
826     std::string options_substring                = "CAPACITY:      ";
827     options_substring                           += std::to_string(this->capacity_kW);
828     options_substring                           += " kW (level ";
829     options_substring                           += std::to_string(this->upgrade_level);
830     options_substring                           += ")\n";
831 
832     options_substring                           += "PRODUCTION:    ";
833     options_substring                           += std::to_string(this->production_MWh);
834     options_substring                           += " MWh\n";
835 
836     options_substring                           += "DISPATCHABLE:  ";
837     options_substring                           += std::to_string(this->dispatchable_MWh);
838     options_substring                           += " MWh\n";
839 
840     options_substring                           += "HEALTH:      ";
841     options_substring                           += std::to_string(this->health);
842     options_substring                           += "/100";
843 
844     if (this->health <= 0) {
845         options_substring                       += " ** BROKEN! **\n";
```

```
846     }
847
848     else {
849         options_substring                       += "\n";
850     }
851
852     options_substring                           += "                                    \n";
853     options_substring                           += "   **** SOLAR PV OPTIONS ****   \n";
854     options_substring                           += "                                    \n";
855
856     if (this->is_broken) {
857         options_substring                       += "     [R]:  REPAIR (";
858         options_substring                       += std::to_string(SOLAR_PV_BUILD_COST);
859         options_substring                       += " K)\n";
860     }
861
862     else {
863         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
864     }
865
866     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
867     options_substring                           += "HOLD [P]:  SCRAP (";
868     options_substring                           += std::to_string(SCRAP_COST);
869     options_substring                           += " K)";
870
871     return options_substring;
872 }   /* getTileOptionsSubstring() */
```

4.11.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
992 {
993     TileImprovement :: processEvent();
994
995     if (this->event_ptr->type == sf::Event::KeyPressed) {
996         this->__handleKeyPressEvents();
997     }
998
999     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1000         this->__handleMouseButtonEvents();
1001     }
1002
1003     return;
1004 }   /* processEvent() */
```

4.11.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1019 {
1020     TileImprovement :: processMessage();
1021
1022     //...
1023
1024     return;
1025 }   /* 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.

```
889 {
890     TileImprovement :: setIsSelected(is_selected);
891 
892     if (this->is_running and this->is_selected) {
893         this->assets_manager_ptr->getSound("solar hum")->play();
894     }
895 
896     return;
897 }   /* setIsSelected() */
```

4.11.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

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

4.11.4 Member Data Documentation

4.11.4.1 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.11.4.2 capacity_kW

```
int SolarPV::capacity_kW
```

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

4.11.4.3 dispatch_MWh

```
int SolarPV::dispatch_MWh
```

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

4.11.4.4 dispatch_vec_MWh

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

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

4.11.4.5 dispatchable_MWh

```
int SolarPV::dispatchable_MWh
```

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

4.11.4.6 max_daily_production_MWh

```
double SolarPV::max_daily_production_MWh
```

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

4.11.4.7 production_MWh

```
int SolarPV::production_MWh
```

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

4.11.4.8 production_vec_MWh

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

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

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

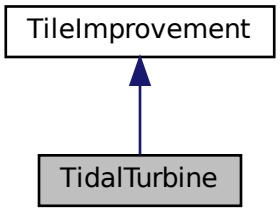
- header/SolarPV.h
- source/SolarPV.cpp

4.12 TidalTurbine Class Reference

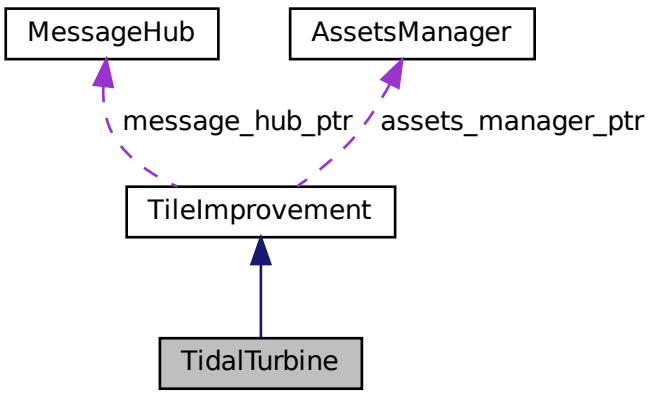
A settlement class (child class of TileImprovement).

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

Inheritance diagram for TidalTurbine:



Collaboration diagram for TidalTurbine:



Public Member Functions

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

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

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

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

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

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

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

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

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

  Destructor for the TidalTurbine class.

Public Attributes

- int capacity_kW

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

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

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

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

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

  The rotation rate of the rotor.
- double bobbing_y

  The bobbing extent of the tidal turbine.
- std::vector< double > capacity_factor_vec

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

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

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

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

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

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

  Helper method to repair the tidal turbine.
- void __computeCapacityFactors (void)

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

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

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

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

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

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

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.12.1 Detailed Description

A settlement class (child class of TileImprovement).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TidalTurbine()

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

Constructor for the TidalTurbine class.

Ref: Wikipedia [2023]

Parameters

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

```
761   :
762 TileImprovement(
763     position_x,
764     position_y,
765     tile_resource,
766     event_ptr,
767     render_window_ptr,
768     assets_manager_ptr,
769     message_hub_ptr
770 )
771 {
772     //  1. set attributes
773
774     //  1.1. private
775     //...
776
777     //  1.2. public
778     this->tile_improvement_type = TileImprovementType :: TIDAL_TURBINE;
779
780     this->is_running = false;
781
782     this->health = 100;
783
784     this->capacity_kW = 100;
785     this->upgrade_level = 1;
786
787     this->storage_kWh = 0;
788     this->storage_level = 0;
789
790     this->production_MWh = 0;
791     this->dispatch_MWh = 0;
792     this->dispatchable_MWh = 0;
793
794     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
795
796     this->rotor_drotation = 64 * SECONDS_PER_FRAME;
797     this->bobbing_y = 4;
798
799     this->capacity_factor_vec.resize(30, 0);
800     this->production_vec_MWh.resize(30, 0);
801     this->dispatch_vec_MWh.resize(30, 0);
802
803     this->tile_improvement_string = "TIDAL TURBINE";
804
805     this->__setUpTileImprovementSpriteAnimated();
806     this->__computeCapacityFactors();
807     this->update();
808
809     std::cout « "TidalTurbine constructed at " « this « std::endl;
810
811     return;
812 }   /* TidalTurbine() */
```

4.12.2.2 ~TidalTurbine()

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

Destructor for the TidalTurbine class.

```
1178 {
1179     std::cout « "TidalTurbine at " « this « " destroyed" « std::endl;
1180
1181     return;
1182 }   /* ~TidalTurbine() */
```

4.12.3 Member Function Documentation

4.12.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.12.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
307 {
308     if (this->is_broken) {
309         return;
310     }
311 
312     for (int i = 0; i < 30; i++) {
313         this->capacity_factor_vec[i] =
314             this->tile_resource_scalar * DAILY_TIDAL_CAPACITY_FACTOR;
315     }
316 
317     return;
318 }   /* __computeCapacityFactors() */
```

4.12.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
366 {
367     if (this->is_broken) {
368         this->dispatchable_MWh = 0;
369         return;
370     }
371 
372     double stored_energy_MWh = 0;
373     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
374 
375     double demand_MWh = 0;
376     double production_MWh = 0;
377     double dispatchable_MWh = 0;
378     double difference_MWh = 0;
379 
380     double room_MWh = 0;
```

```
381
382     for (int i = 0; i < 30; i++) {
383         demand_MWh = this->demand_vec_MWh[i];
384         production_MWh = this->production_vec_MWh[i];
385
386         if (production_MWh <= demand_MWh) {
387             this->dispatch_vec_MWh[i] = production_MWh;
388             dispatchable_MWh += this->dispatch_vec_MWh[i];
389
390             difference_MWh = demand_MWh - production_MWh;
391
392             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
393                 if (difference_MWh > stored_energy_MWh) {
394                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
395                     dispatchable_MWh += stored_energy_MWh;
396                     stored_energy_MWh = 0;
397                 }
398
399                 else {
400                     this->dispatch_vec_MWh[i] += difference_MWh;
401                     dispatchable_MWh += difference_MWh;
402                     stored_energy_MWh -= difference_MWh;
403                 }
404             }
405         }
406
407         else {
408             this->dispatch_vec_MWh[i] = demand_MWh;
409             dispatchable_MWh += this->dispatch_vec_MWh[i];
410
411             difference_MWh = production_MWh - demand_MWh;
412
413             if (
414                 (storage_capacity_MWh > 0) and
415                 (stored_energy_MWh < storage_capacity_MWh)
416             ) {
417                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
418
419                 if (difference_MWh > room_MWh) {
420                     stored_energy_MWh += room_MWh;
421                 }
422
423                 else {
424                     stored_energy_MWh += difference_MWh;
425                 }
426             }
427         }
428     }
429
430     this->dispatchable_MWh = round(dispatchable_MWh);
431
432     if (this->dispatch_MWh != this->dispatchable_MWh) {
433         this->dispatch_MWh = this->dispatchable_MWh;
434     }
435
436     return;
437 }   /* __computeDispatch() */
```

4.12.3.4 __computeProduction()

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

Helper method to compute production values.

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

```
347
348     this->production_MWh = round(production_MWh);
349
350     return;
351 }   /* __computeProduction() */
```

4.12.3.5 __computeProductionCosts()

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

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

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

4.12.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

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

```
157
158     production_text.setOrigin(production_text.getLocalBounds().width / 2,0);
159     production_text.setFillColor(MONOCHROME_TEXT_GREEN);
160
161     production_text.setPosition(400 + 30, 400 - 45);
162
163     this->render_window_ptr->draw(production_text);
164
165     return;
166 }   /* __drawProductionMenu() */
```

4.12.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
578 {
579     //  1. draw power capacity upgrade sprite
580     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
581         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
582         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 8);
583
584         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
585         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
586
587         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
588         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
589
590         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
591         this->tile_improvement_sprite_animated[i].setRotation(0);
592
593         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
594
595         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
596         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
597         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
598         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
599     }
600
601     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
602
603
604     //  2. draw power capacity upgrade text
605     //                  16 char line = "                \n"
606     std::string power_upgrade_string = "POWER CAPACITY  \n";
607     power_upgrade_string            += "                \n";
608
609     power_upgrade_string            += "CAPACITY:  ";
610     power_upgrade_string            += std::to_string(this->capacity_kW);
611     power_upgrade_string            += " kW\n";
612
613     power_upgrade_string            += "LEVEL:     ";
614     power_upgrade_string            += std::to_string(this->upgrade_level);
615     power_upgrade_string            += "\n\n";
616
617     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
618         power_upgrade_string        += "[W]: + 100 kW (";
619         power_upgrade_string        +=  std::to_string(TIDAL_TURBINE_BUILD_COST);
620         power_upgrade_string        += " K)\n";
621     }
622
623     else {
624         power_upgrade_string        += " * MAX LEVEL *  \n";
625     }
626
627     sf::Text power_upgrade_text = sf::Text(
628         power_upgrade_string,
629         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
630         16
631     );
632
633     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
634     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
635     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
636
637     this->render_window_ptr->draw(power_upgrade_text);
638
```

```
639
640     //  3. draw energy capacity (storage) upgrade sprite
641     this->render_window_ptr->draw(this->storage_upgrade_sprite);
642     this->render_window_ptr->draw(this->upgrade_plus_sprite);
643
644
645     //  4. draw energy capacity (storage) upgrade text
646     //                   16 char line = "                \n"
647     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
648     energy_upgrade_string            += "                \n";
649
650     energy_upgrade_string            += "CAPACITY:  ";
651     energy_upgrade_string            += std::to_string(this->storage_level * 200);
652     energy_upgrade_string            += " kWh\n";
653
654     energy_upgrade_string            += "LEVEL:     ";
655     energy_upgrade_string            += std::to_string(this->storage_level);
656     energy_upgrade_string            += "\n\n";
657
658     if (this->storage_level < MAX_STORAGE_LEVELS) {
659         energy_upgrade_string        += "[D]: + 200 kWh (";
660         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
661         energy_upgrade_string        += " K)\n";
662     }
663
664     else {
665         energy_upgrade_string += " * MAX LEVEL *  \n";
666     }
667
668     sf::Text energy_upgrade_text = sf::Text(
669         energy_upgrade_string,
670         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
671         16
672     );
673
674     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
675     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
676     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
677
678     this->render_window_ptr->draw(energy_upgrade_text);
679
680     return;
681 }   /* __drawUpgradeOptions() */
```

4.12.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
452 {
453     if (this->just_built) {
454         return;
455     }
456
457     switch (this->event_ptr->key.code) {
458         case (sf::Keyboard::U): {
459             this->__openUpgradeMenu();
460
461             break;
462         }
463
464
465         case (sf::Keyboard::W): {
466             if (this->production_menu_open) {
467                 this->dispatch_MWh++;
468
469                 if (this->dispatch_MWh > this->dispatchable_MWh) {
470                     this->dispatch_MWh = 0;
471                 }
472
473                 this->__computeProductionCosts();
474                 this->assets_manager_ptr->getSound("interface click")->play();
475             }
476
477             else if (this->upgrade_menu_open) {
478                 this->__upgradePowerCapacity();
479             }
```

```
480
481             break;
482         }
483
484
485         case (sf::Keyboard::S): {
486             if (this->production_menu_open) {
487                 this->dispatch_MWh--;
488
489                 if (this->dispatch_MWh < 0) {
490                     this->dispatch_MWh = this->dispatchable_MWh;
491                 }
492
493                 this->__computeProductionCosts();
494                 this->assets_manager_ptr->getSound("interface click")->play();
495             }
496
497             break;
498         }
499
500
501         case (sf::Keyboard::D): {
502             if (this->upgrade_menu_open) {
503                 this->__upgradeStorageCapacity();
504                 this->__computeProduction();
505                 this->__computeDispatch();
506             }
507
508             break;
509         }
510
511
512         default: {
513             // do nothing!
514
515             break;
516         }
517     }
518
519     return;
520 }   /* __handleKeyPressEvents() */
```

4.12.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
535 {
536     if (this->just_built) {
537         return;
538     }
539
540     switch (this->event_ptr->mouseButton.button) {
541         case (sf::Mouse::Left): {
542             //...
543
544             break;
545         }
546
547
548         case (sf::Mouse::Right): {
549             //...
550
551             break;
552         }
553
554
555         default: {
556             // do nothing!
557
558             break;
559         }
560     }
561
562     return;
563 }   /* __handleMouseButtonEvents() */
```

4.12.3.10 __repair()

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

Helper method to repair the tidal turbine.

Reimplemented from TileImprovement.

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

4.12.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
696 {
697     Message improvement_state_message;
698 
699     improvement_state_message.channel = GAME_CHANNEL;
700     improvement_state_message.subject = "improvement state";
701 
702     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
703     improvement_state_message.int_payload["operation_maintenance_cost"] =
704         this->operation_maintenance_cost;
705 
706     this->message_hub_ptr->sendMessage(improvement_state_message);
707 
708     std::cout « "Improvement state message sent by " « this « std::endl;
709 
710     return;
711 }   /* __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.

```
918 {
919     //  1. send improvement state message
920     this->__sendImprovementStateMessage();
921 
922     //  2. update
923     this->__computeCapacityFactors();
924     this->update();
925 
926     //  3. handle start/stop
927     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
928         this->is_running = true;
929     }
930 
931     else if (this->is_running and (this->dispatch_MWh <= 0)) {
932         this->is_running = false;
933     }
934 
935     //  4. handle equipment health and breakdowns
936     if (this->is_running) {
937         this->health--;
938 
939         if (this->health <= 50) {
940             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
941 
942             if ((double)rand() / RAND_MAX <= breakdown_prob) {
943                 this->health = 0;
944             }
945         }
946 
947         if (this->health <= 0) {
948             this->__breakdown();
949         }
950     }
951 
952     //  5. send tile state request (if selected)
953     if (this->is_selected) {
954         this->__sendTileStateRequest();
955     }
956 
957     return;
958 }   /* 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.

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

```
1070                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1071                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1072                )
1073            );
1074        }
1075
1076        this->upgrade_frame++;
1077    }
1078
1079    if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1080        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1081            this->tile_improvement_sprite_animated[i].setColor(
1082                sf::Color(255,255,255,255)
1083            );
1084
1085            this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1086        }
1087
1088        this->just_upgraded = false;
1089        this->upgrade_frame = 0;
1090    }
1091
1092
1093    //  3. handle bobbing
1094    for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1095        this->tile_improvement_sprite_animated[i].setPosition(
1096            this->position_x,
1097            this->position_y + this->bobbing_y * cos(
1098                (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1099            )
1100        );
1101    }
1102
1103
1104    //  4. draw first element of animated sprite
1105    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1106
1107
1108    //  5. draw second element of animated sprite
1109    if (this->is_running) {
1110        this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1111    }
1112
1113    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1114
1115
1116    //  6. draw storage upgrades
1117    for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1118        this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1119    }
1120
1121
1122    //  7. handle dispatch illustration
1123    if (this->dispatch_MWh > 0) {
1124        this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1125        this->__drawDispatch();
1126    }
1127
1128
1129    //  8. draw production menu
1130    if (this->production_menu_open) {
1131        this->render_window_ptr->draw(this->production_menu_backing);
1132        this->render_window_ptr->draw(this->production_menu_backing_text);
1133
1134        this->__drawProductionMenu();
1135    }
1136
1137
1138    //  9. draw upgrade menu
1139    if (this->upgrade_menu_open) {
1140        this->render_window_ptr->draw(this->upgrade_menu_backing);
1141        this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1142
1143        this->__drawUpgradeOptions();
1144    }
1145
1146
1147    //  10. handle broken effects
1148    if (this->is_broken) {
1149        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1150            this->tile_improvement_sprite_animated[i].setColor(
1151                sf::Color(
1152                    255,
1153                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1154                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1155                    255
1156                )
```

```
1157             );
1158         }
1159     }
1160
1161     this->frame++;
1162     return;
1163 }   /* 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.

```
829 {
830     //                  32 char x 17 line console "--------------------------------\n";
831     std::string options_substring                = "CAPACITY:      ";
832     options_substring                           += std::to_string(this->capacity_kW);
833     options_substring                           += " kW (level ";
834     options_substring                           += std::to_string(this->upgrade_level);
835     options_substring                           += ")\n";
836
837     options_substring                           += "PRODUCTION:    ";
838     options_substring                           += std::to_string(this->production_MWh);
839     options_substring                           += " MWh\n";
840
841     options_substring                           += "DISPATCHABLE:  ";
842     options_substring                           += std::to_string(this->dispatchable_MWh);
843     options_substring                           += " MWh\n";
844
845     options_substring                           += "HEALTH:      ";
846     options_substring                           += std::to_string(this->health);
847     options_substring                           += "/100";
848
849     if (this->health <= 0) {
850         options_substring                       += " ** BROKEN! **\n";
851     }
852
853     else {
854         options_substring                       += "\n";
855     }
856
857     options_substring                           += "                                \n";
858     options_substring                           += "**** TIDAL TURBINE OPTIONS **** \n";
859     options_substring                           += "                                \n";
860
861     if (this->is_broken) {
862         options_substring                       += "     [R]:  REPAIR (";
863         options_substring                       += std::to_string(TIDAL_TURBINE_BUILD_COST);
864         options_substring                       += " K)\n";
865     }
866
867     else {
868         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
869     }
870
871     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
872     options_substring                           += "HOLD [P]:  SCRAP (";
873     options_substring                           += std::to_string(SCRAP_COST);
874     options_substring                           += " K)";
875
876     return options_substring;
877 }   /* getTileOptionsSubstring() */
```

4.12.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
998 {
999     TileImprovement :: processEvent();
1000
1001     if (this->event_ptr->type == sf::Event::KeyPressed) {
1002         this->__handleKeyPressEvents();
1003     }
1004
1005     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1006         this->__handleMouseButtonEvents();
1007     }
1008
1009     return;
1010 }   /* processEvent() */
```

4.12.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1025 {
1026     TileImprovement :: processMessage();
1027
1028     //...
1029
1030     return;
1031 }   /* 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.

```
894 {
895     TileImprovement :: setIsSelected(is_selected);
896
897     if (this->is_running and this->is_selected) {
898         this->assets_manager_ptr->getSound("water flow")->play();
899     }
900
```

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

4.12.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
973 {
974     this->__computeProduction();
975     this->__computeProductionCosts();
976     this->__computeDispatch();
977 
978     if (this->is_selected) {
979         this->__sendTileStateRequest();
980     }
981 
982     return;
983 }   /* update() */
```

4.12.4 Member Data Documentation

4.12.4.1 bobbing_y

```
double TidalTurbine::bobbing_y
```

The bobbing extent of the tidal turbine.

4.12.4.2 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.12.4.3 capacity_kW

```
int TidalTurbine::capacity_kW
```

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

4.12.4.4 dispatch_MWh

```
int TidalTurbine::dispatch_MWh
```

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

4.12.4.5 dispatch_vec_MWh

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

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

4.12.4.6 dispatchable_MWh

```
int TidalTurbine::dispatchable_MWh
```

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

4.12.4.7 max_daily_production_MWh

```
double TidalTurbine::max_daily_production_MWh
```

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

4.12.4.8 production_MWh

```
int TidalTurbine::production_MWh
```

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

4.12.4.9 production_vec_MWh

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

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

4.12.4.10 rotor_drotation

```
double TidalTurbine::rotor_drotation
```

The rotation rate of the rotor.

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

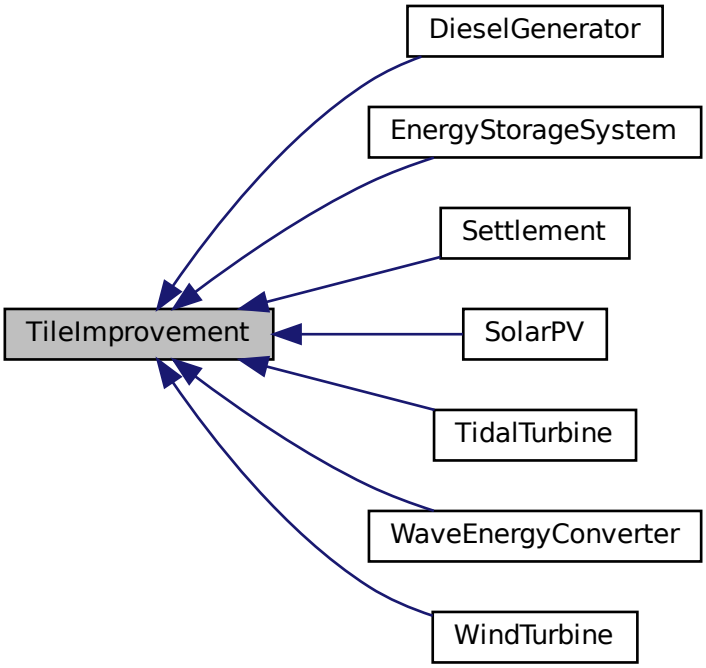
- header/TidalTurbine.h
- source/TidalTurbine.cpp

4.13 TileImprovement Class Reference

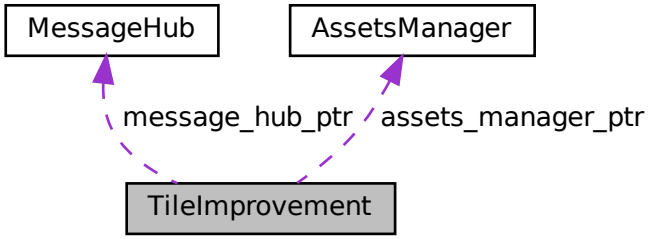
A base class for the tile improvement hierarchy.

```
#include <TileImprovement.h>
```

Inheritance diagram for TileImprovement:



Collaboration diagram for TileImprovement:



Public Member Functions

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

  Constructor for the TileImprovement class.
- virtual void setIsSelected (bool)

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

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

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

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

  Destructor for the TileImprovement class.

Public Attributes

- TileImprovementType tile_improvement_type

  The type of the tile improvement.
- bool is_running

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

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

  A boolean which indicates that the improvement was just built.
- bool just_upgraded

  A boolean which indicates that the improvement was just upgraded.
- bool production_menu_open

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

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

- bool is_broken

  A boolean which indicated whether or not improvement is broken.
- unsigned long long int frame

  The current frame of this object.
- int credits

  The current balance of credits.
- int month

  The current month of play.
- int demand_MWh

  The current demand [MWh].
- int health

  The health of the improvement.
- int upgrade_level

  The upgrade level of the improvement.
- int upgrade_frame

  The frame of the upgrade animation.
- int storage_kWh

  The rated energy capacity [kWh] of the storage.
- int storage_level

  The level of storage installed alongside the tile improvement.
- int operation_maintenance_cost

  The operation and maintenance costs for this turn.
- int tile_resource

  The renewable resource quality of the tile.
- double tile_resource_scalar

  A scalar associated with the renewable resource quality.
- double position_x

  The x position of the tile improvement.
- double position_y

  The y position of the tile improvement.
- std::vector< double > demand_vec_MWh

  A vector of daily demands [MWh] for the current month.
- std::string game_phase

  The current phase of the game.
- std::string tile_improvement_string

  A string representation of the tile improvement type.
- sf::Sprite tile_improvement_sprite_static

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

  An animated sprite, for the ContextMenu visual screen.
- sf::RectangleShape production_menu_backing

  A backing for the production menu.
- sf::Text production_menu_backing_text

  Text for the production menu backing.
- sf::RectangleShape upgrade_menu_backing

  A backing for the upgrade menu.
- sf::Text upgrade_menu_backing_text

  Text for the upgrade menu backing.
- sf::Sprite storage_upgrade_sprite

  A sprite for illustrating storage (in upgrade menu).
- std::vector< sf::Sprite > storage_upgrade_sprite_vec

A vector of sprites for illustrating the storage upgrade level (on tile).

- sf::Sprite upgrade_arrow_sprite

  An upgrade arrow sprite.
- sf::Sprite upgrade_plus_sprite

  An upgrade plus sprite.
- sf::CircleShape dispatch_backing

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

  Text for illustrating dispatch.

Protected Member Functions

- void __setUpProductionMenu (void)

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

  Helper method to set up and position upgrade menu assets (drawable).
- void __setUpDispatchIllustration (void)

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

  Helper method to upgrade storage capacity.
- void __handleKeyPressEvents (void)

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

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

  Helper method to open the production menu.
- void __closeProductionMenu (void)

  Helper method to close the production menu.
- void __breakdown (void)

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

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

  Helper method to open the upgrade menu.
- void __closeUpgradeMenu (void)

  Helper method to close the build menu.
- void __sendTileStateRequest (void)

  Helper method to format and send a request for the parent HexTile to send a tile state message.
- void __sendGameStateRequest (void)

  Helper method to format and send a game state request (message).
- void __sendCreditsSpentMessage (int)

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

  Helper method to format and send an insufficient credits message.
- void __drawDispatch (void)

  Helper method to draw dispatch illustration.

Protected Attributes

- sf::Event * event_ptr

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

  A pointer to the render window.
- AssetsManager * assets_manager_ptr

  A pointer to the assets manager.
- MessageHub * message_hub_ptr

  A pointer to the message hub.

4.13.1 Detailed Description

A base class for the tile improvement hierarchy.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TileImprovement()

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

Constructor for the TileImprovement class.

Ref: Wikipedia [2023]

Parameters

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

```
727 {
728     //  1. set attributes
729
730     //  1.1. protected
731     this->event_ptr = event_ptr;
732     this->render_window_ptr = render_window_ptr;
733
```

```
734     this->assets_manager_ptr = assets_manager_ptr;
735     this->message_hub_ptr = message_hub_ptr;
736 
737     //  1.2. public
738     this->is_selected = true;
739     this->just_built = true;
740     this->production_menu_open = false;
741     this->upgrade_menu_open = false;
742     this->is_broken = false;
743 
744     this->just_upgraded = false;
745     this->upgrade_frame = 0;
746 
747     this->frame = 0;
748     this->credits = 0;
749     this->month = 1;
750     this->demand_MWh = 0;
751 
752     this->demand_vec_MWh.resize(30, 0);
753 
754     this->operation_maintenance_cost = 0;
755 
756     this->tile_resource = tile_resource;
757 
758     switch (this->tile_resource) {
759         case (0): {
760             this->tile_resource_scalar = 0.85;
761 
762             break;
763         }
764 
765 
766         case (1): {
767             this->tile_resource_scalar = 0.925;
768 
769             break;
770         }
771 
772 
773         case (2): {
774             this->tile_resource_scalar = 1;
775 
776             break;
777         }
778 
779 
780         case (3): {
781             this->tile_resource_scalar = 1.075;
782 
783             break;
784         }
785 
786 
787         case (4): {
788             this->tile_resource_scalar = 1.15;
789 
790             break;
791         }
792 
793 
794         default: {
795             this->tile_resource_scalar = 1;
796         }
797     }
798 
799     this->position_x = position_x;
800     this->position_y = position_y;
801 
802     this->game_phase = "build settlement";
803 
804     this->__setUpProductionMenu();
805     this->__setUpUpgradeMenu();
806     this->__setUpDispatchIllustration();
807 
808     std::cout « "TileImprovement constructed at " « this « std::endl;
809 
810     return;
811 }   /* TileImprovement() */
```

4.13.2.2 ~TileImprovement()

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

```
            void  )  [virtual]
```

Destructor for the TileImprovement class.

```
1044 {
1045     std::cout « "TileImprovement at " « this « " destroyed" « std::endl;
1046
1047     return;
1048 }   /* ~TileImprovement() */
```

4.13.3 Member Function Documentation

4.13.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

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

4.13.3.4 __drawDispatch()

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

Helper method to draw dispatch illustration.

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

4.13.3.5 __handleKeyPressEvents()

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

Helper method to handle key press events.

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

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

4.13.3.6 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

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

4.13.3.7 __openProductionMenu()

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

Helper method to open the production menu.

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

4.13.3.8 __openUpgradeMenu()

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

Helper method to open the upgrade menu.

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

4.13.3.9 __repair()

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

Helper method to repair a tile improvement.

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

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

4.13.3.10 __sendCreditsSpentMessage()

```
void TileImprovement::__sendCreditsSpentMessage (
            int credits_spent )  [protected]
```

Helper method to format and send a credits spent message.

Parameters

credits_spent	The number of credits that were spent.

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

4.13.3.11 __sendGameStateRequest()

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

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

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

4.13.3.12 __sendInsufficientCreditsMessage()

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

Helper method to format and send an insufficient credits message.

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

4.13.3.13 __sendTileStateRequest()

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

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

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

4.13.3.14 __setUpDispatchIllustration()

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

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

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

4.13.3.15 __setUpProductionMenu()

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

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

```
68 {
69     //  1. set up and place production menu backing and text
70     this->production_menu_backing.setSize(sf::Vector2f(400, 256));
71     this->production_menu_backing.setOrigin(200, 128);
72     this->production_menu_backing.setPosition(400, 400);
73     this->production_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
74     this->production_menu_backing.setOutlineColor(MENU_FRAME_GREY);
75     this->production_menu_backing.setOutlineThickness(4);
76 
77     this->production_menu_backing_text.setString("**** PRODUCTION MENU ****");
78     this->production_menu_backing_text.setFont(
79         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
80     );
81     this->production_menu_backing_text.setCharacterSize(16);
82     this->production_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
83     this->production_menu_backing_text.setOrigin(
84         this->production_menu_backing_text.getLocalBounds().width / 2, 0
85     );
86     this->production_menu_backing_text.setPosition(400, 400 - 128 + 4);
87 
88     return;
89 }   /* __setUpProductionMenu() */
```

4.13.3.16 __setUpUpgradeMenu()

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

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

```
104 {
105     //  1. set up and place upgrade menu backing and text
106     this->upgrade_menu_backing.setSize(sf::Vector2f(400, 256));
107     this->upgrade_menu_backing.setOrigin(200, 128);
108     this->upgrade_menu_backing.setPosition(400, 400);
109     this->upgrade_menu_backing.setFillColor(MONOCHROME_SCREEN_BACKGROUND);
110     this->upgrade_menu_backing.setOutlineColor(MENU_FRAME_GREY);
111     this->upgrade_menu_backing.setOutlineThickness(4);
112 
113     this->upgrade_menu_backing_text.setString("**** UPGRADE MENU ****");
114     this->upgrade_menu_backing_text.setFont(
115         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220"))
116     );
117     this->upgrade_menu_backing_text.setCharacterSize(16);
118     this->upgrade_menu_backing_text.setFillColor(MONOCHROME_TEXT_GREEN);
119     this->upgrade_menu_backing_text.setOrigin(
120         this->upgrade_menu_backing_text.getLocalBounds().width / 2, 0
121     );
122     this->upgrade_menu_backing_text.setPosition(400, 400 - 128 + 4);
123 
124 
125     //  2. set up and place storage upgrade sprite (with upgrade plus)
126     this->storage_upgrade_sprite = sf::Sprite(
127         *(this->assets_manager_ptr->getTexture("energy storage system"))
128     );
129 
130     this->storage_upgrade_sprite.setOrigin(
131         this->storage_upgrade_sprite.getLocalBounds().width / 2,
132         this->storage_upgrade_sprite.getLocalBounds().height
133     );
134 
135     this->storage_upgrade_sprite.setPosition(400 + 100, 400 - 32);
136 
137     this->upgrade_plus_sprite = sf::Sprite(
138         *(this->assets_manager_ptr->getTexture("upgrade plus"))
139     );
140 
141     this->upgrade_plus_sprite.setOrigin(
142         this->upgrade_plus_sprite.getLocalBounds().width / 2,
143         this->upgrade_plus_sprite.getLocalBounds().height / 2
144     );
145 
146     this->upgrade_plus_sprite.setPosition(400 + 130, 400 - 64);
147 
148 
149     //  3. set up and place upgrade arrow sprite
150     this->upgrade_arrow_sprite = sf::Sprite(
151         *(this->assets_manager_ptr->getTexture("upgrade arrow"))
152     );
153 
154     this->upgrade_arrow_sprite.setOrigin(
155         this->upgrade_arrow_sprite.getLocalBounds().width / 2,
156         this->upgrade_arrow_sprite.getLocalBounds().height / 2
157     );
158 
159     this->upgrade_arrow_sprite.setPosition(400 - 64, 400 - 64);
160 
161 
162     return;
163 }   /* __setUpUpgradeMenu() */
```

4.13.3.17 __upgradeStorageCapacity()

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

Helper method to upgrade storage capacity.

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

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

4.13.3.18 advanceTurn()

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

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

```
191 {return;}
```

4.13.3.19 draw()

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

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

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

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

```
926
927         if (
928             (alpha >= 255) or
929             (this->tile_improvement_sprite_static.getPosition().y >= this->position_y + 12)
930         ) {
931             this->tile_improvement_sprite_static.setColor(
932                 sf::Color(255, 255, 255, 255)
933             );
934
935             this->tile_improvement_sprite_static.setPosition(
936                 this->position_x,
937                 this->position_y + 12
938             );
939
940             this->just_built = false;
941             this->assets_manager_ptr->getSound("place improvement")->play();
942         }
943
944         this->render_window_ptr->draw(this->tile_improvement_sprite_static);
945     }
946
947
948     else {
949         int alpha = 0;
950
951         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
952             alpha = this->tile_improvement_sprite_animated[i].getColor().a;
953
954             alpha += 0.08 * FRAMES_PER_SECOND;
955
956             this->tile_improvement_sprite_animated[i].setColor(
957                 sf::Color(255, 255, 255, alpha)
958             );
959
960             this->tile_improvement_sprite_animated[i].move(0, 50 * SECONDS_PER_FRAME);
961
962             if (
963                 (alpha >= 255) or
964                 (this->tile_improvement_sprite_animated[i].getPosition().y >= this->position_y + 12)
965             ) {
966                 this->tile_improvement_sprite_animated[i].setColor(
967                     sf::Color(255, 255, 255, 255)
968                 );
969
970                 this->tile_improvement_sprite_animated[i].setPosition(
971                     this->position_x,
972                     this->position_y + 12
973                 );
974             }
975
976             this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
977         }
978
979         if (
980             (alpha >= 255) or
981             (this->tile_improvement_sprite_animated[0].getPosition().y >= this->position_y + 12)
982         ) {
983             this->just_built = false;
984             this->assets_manager_ptr->getSound("place improvement")->play();
985
986             switch (this->tile_improvement_type) {
987                 case (TileImprovementType :: WIND_TURBINE): {
988                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
989                         this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
990                         this->tile_improvement_sprite_animated[i].move(0, -32);
991                     }
992
993                     break;
994                 }
995
996
997                 case (TileImprovementType :: TIDAL_TURBINE): {
998                     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
999                         this->tile_improvement_sprite_animated[i].setOrigin(32, 45);
1000                          this->tile_improvement_sprite_animated[i].move(0, -19);
1001                      }
1002
1003                      break;
1004                  }
1005
1006
1007                  case (TileImprovementType :: WAVE_ENERGY_CONVERTER): {
1008                      for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1009                          this->tile_improvement_sprite_animated[i].setOrigin(32, 32);
1010                          this->tile_improvement_sprite_animated[i].move(0, -32);
1011                      }
1012
```

```
1013                     break;
1014                 }
1015
1016
1017                 default: {
1018                     // do nothing!
1019
1020                     break;
1021                 }
1022             }
1023         }
1024     }
1025
1026
1027     this->frame++;
1028     return;
1029 }   /* draw() */
```

4.13.3.20 getTileOptionsSubstring()

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

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

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

4.13.3.21 processEvent()

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

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

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

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

4.13.3.22 processMessage()

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

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

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

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

4.13.3.23 setIsSelected()

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

Method to set the is selected attribute.

Parameters

is_selected	The value to set the is selected attribute to.

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

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

4.13.3.24 update()

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

Reimplemented in WindTurbine, WaveEnergyConverter, TidalTurbine, and SolarPV.
193 {return;}

4.13.4 Member Data Documentation

4.13.4.1 assets_manager_ptr

```
AssetsManager* TileImprovement::assets_manager_ptr  [protected]
```

A pointer to the assets manager.

4.13.4.2 credits

```
int TileImprovement::credits
```

The current balance of credits.

4.13.4.3 demand_MWh

```
int TileImprovement::demand_MWh
```

The current demand [MWh].

4.13.4.4 demand_vec_MWh

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

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

4.13.4.5 dispatch_backing

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

A backing circle for dispatch text illustration.

4.13.4.6 dispatch_text

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

Text for illustrating dispatch.

4.13.4.7 event_ptr

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

A pointer to the event class.

4.13.4.8 frame

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

The current frame of this object.

4.13.4.9 game_phase

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

The current phase of the game.

4.13.4.10 health

```
int TileImprovement::health
```

The health of the improvement.

4.13.4.11 is_broken

```
bool TileImprovement::is_broken
```

A boolean which indicated whether or not improvement is broken.

4.13.4.12 is_running

```
bool TileImprovement::is_running
```

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

4.13.4.13 is_selected

```
bool TileImprovement::is_selected
```

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

4.13.4.14 just_built

```
bool TileImprovement::just_built
```

A boolean which indicates that the improvement was just built.

4.13.4.15 just_upgraded

```
bool TileImprovement::just_upgraded
```

A boolean which indicates that the improvement was just upgraded.

4.13.4.16 message_hub_ptr

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

A pointer to the message hub.

4.13.4.17 month

```
int TileImprovement::month
```

The current month of play.

4.13.4.18 operation_maintenance_cost

```
int TileImprovement::operation_maintenance_cost
```

The operation and maintenance costs for this turn.

4.13.4.19 position_x

```
double TileImprovement::position_x
```

The x position of the tile improvement.

4.13.4.20 position_y

```
double TileImprovement::position_y
```

The y position of the tile improvement.

4.13.4.21 production_menu_backing

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

A backing for the production menu.

4.13.4.22 production_menu_backing_text

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

Text for the production menu backing.

4.13.4.23 production_menu_open

```
bool TileImprovement::production_menu_open
```

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

4.13.4.24 render_window_ptr

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

A pointer to the render window.

4.13.4.25 storage_kWh

```
int TileImprovement::storage_kWh
```

The rated energy capacity [kWh] of the storage.

4.13.4.26 storage_level

```
int TileImprovement::storage_level
```

The level of storage installed alongside the tile improvement.

4.13.4.27 storage_upgrade_sprite

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

A sprite for illustrating storage (in upgrade menu).

4.13.4.28 storage_upgrade_sprite_vec

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

A vector of sprites for illustrating the storage upgrade level (on tile).

4.13.4.29 tile_improvement_sprite_animated

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

An animated sprite, for the ContextMenu visual screen.

4.13.4.30 tile_improvement_sprite_static

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

A static sprite, for decorating the tile.

4.13.4.31 tile_improvement_string

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

A string representation of the tile improvement type.

4.13.4.32 tile_improvement_type

```
TileImprovementType TileImprovement::tile_improvement_type
```

The type of the tile improvement.

4.13.4.33 tile_resource

```
int TileImprovement::tile_resource
```

The renewable resource quality of the tile.

4.13.4.34 tile_resource_scalar

```
double TileImprovement::tile_resource_scalar
```

A scalar associated with the renewable resource quality.

4.13.4.35 upgrade_arrow_sprite

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

An upgrade arrow sprite.

4.13.4.36 upgrade_frame

```
int TileImprovement::upgrade_frame
```

The frame of the upgrade animation.

4.13.4.37 upgrade_level

```
int TileImprovement::upgrade_level
```

The upgrade level of the improvement.

4.13.4.38 upgrade_menu_backing

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

A backing for the upgrade menu.

4.13.4.39 upgrade_menu_backing_text

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

Text for the upgrade menu backing.

4.13.4.40 upgrade_menu_open

```
bool TileImprovement::upgrade_menu_open
```

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

4.13.4.41 upgrade_plus_sprite

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

An upgrade plus sprite.

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

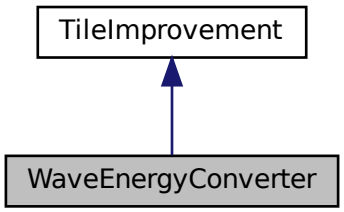
- header/TileImprovement.h
- source/TileImprovement.cpp

4.14 WaveEnergyConverter Class Reference

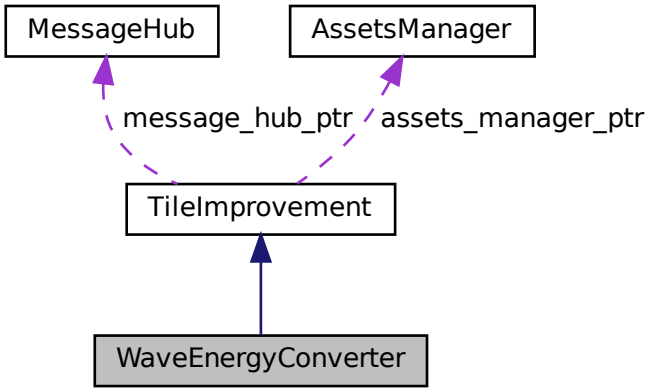
A settlement class (child class of TileImprovement).

`#include <WaveEnergyConverter.h>`

Inheritance diagram for WaveEnergyConverter:



Collaboration diagram for WaveEnergyConverter:



Public Member Functions

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

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

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

  Method to set the is selected attribute.

- void advanceTurn (void)

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

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

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

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

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

  Destructor for the WaveEnergyConverter class.

Public Attributes

- int capacity_kW

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

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

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

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

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

  The bobbing extent of the wave energy converter.
- std::vector< double > capacity_factor_vec

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

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

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

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

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

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

  Helper method to repair the wave energy converter.
- void __computeCapacityFactors (void)

Helper method to compute capacity factors.

- void __computeProduction (void)

  Helper method to compute production values.

- void __computeDispatch (void)

  Helper method to compute dispatch values.

- void __handleKeyPressEvents (void)

  Helper method to handle key press events.

- void __handleMouseButtonEvents (void)

  Helper method to handle mouse button events.

- void __drawUpgradeOptions (void)

  Helper method to set up and draw upgrade options.

- void __sendImprovementStateMessage (void)

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.14.1 Detailed Description

A settlement class (child class of TileImprovement).

4.14.2 Constructor & Destructor Documentation

4.14.2.1 WaveEnergyConverter()

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

Constructor for the WaveEnergyConverter class.

Ref: Wikipedia [2023]

Parameters

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

```
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 :: WAVE_ENERGY_CONVERTER;
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->bobbing_y = 4;
813 
814     this->capacity_factor_vec.resize(30, 0);
815     this->production_vec_MWh.resize(30, 0);
816     this->dispatch_vec_MWh.resize(30, 0);
817 
818     this->tile_improvement_string = "WAVE ENERGY";
819 
820     this->__setUpTileImprovementSpriteAnimated();
821     this->__computeCapacityFactors();
822     this->update();
823 
824     std::cout « "WaveEnergyConverter constructed at " « this « std::endl;
825 
826     return;
827 }   /* WaveEnergyConverter() */
```

4.14.2.2 ~WaveEnergyConverter()

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

Destructor for the WaveEnergyConverter class.

```
1204 {
1205     std::cout « "WaveEnergyConverter at " « this « " destroyed" « std::endl;
1206 
1207     return;
1208 }   /* ~WaveEnergyConverter() */
```

4.14.3 Member Function Documentation

4.14.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.14.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
307 {
308     if (this->is_broken) {
309         return;
310     }
311 
312     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
313     std::default_random_engine generator(seed);
314 
315     double mean =
316         this->tile_resource_scalar * MEAN_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
317 
318     double stdev = STDEV_DAILY_WAVE_CAPACITY_FACTORS[this->month - 1];
319 
320     if (this->tile_resource_scalar > 1) {
321         stdev /= this->tile_resource_scalar;
322     }
323 
324     std::normal_distribution<double> normal_dist(mean, stdev);
325 
326     double capacity_factor = 0;
327 
328     for (int i = 0; i < 30; i++) {
329         capacity_factor = normal_dist(generator);
330 
331         if (capacity_factor < 0) {
332             capacity_factor = 0;
333         }
334 
335         this->capacity_factor_vec[i] = capacity_factor;
336     }
337 
338     return;
339 }   /* __computeCapacityFactors() */
```

4.14.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
387 {
388     if (this->is_broken) {
389         this->dispatchable_MWh = 0;
```

```
390         return;
391     }
392
393     double stored_energy_MWh = 0;
394     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
395
396     double demand_MWh = 0;
397     double production_MWh = 0;
398     double dispatchable_MWh = 0;
399     double difference_MWh = 0;
400
401     double room_MWh = 0;
402
403     for (int i = 0; i < 30; i++) {
404         demand_MWh = this->demand_vec_MWh[i];
405         production_MWh = this->production_vec_MWh[i];
406
407         if (production_MWh <= demand_MWh) {
408             this->dispatch_vec_MWh[i] = production_MWh;
409             dispatchable_MWh += this->dispatch_vec_MWh[i];
410
411             difference_MWh = demand_MWh - production_MWh;
412
413             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
414                 if (difference_MWh > stored_energy_MWh) {
415                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
416                     dispatchable_MWh += stored_energy_MWh;
417                     stored_energy_MWh = 0;
418                 }
419
420                 else {
421                     this->dispatch_vec_MWh[i] += difference_MWh;
422                     dispatchable_MWh += difference_MWh;
423                     stored_energy_MWh -= difference_MWh;
424                 }
425             }
426         }
427
428         else {
429             this->dispatch_vec_MWh[i] = demand_MWh;
430             dispatchable_MWh += this->dispatch_vec_MWh[i];
431
432             difference_MWh = production_MWh - demand_MWh;
433
434             if (
435                 (storage_capacity_MWh > 0) and
436                 (stored_energy_MWh < storage_capacity_MWh)
437             ) {
438                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
439
440                 if (difference_MWh > room_MWh) {
441                     stored_energy_MWh += room_MWh;
442                 }
443
444                 else {
445                     stored_energy_MWh += difference_MWh;
446                 }
447             }
448         }
449     }
450
451     this->dispatchable_MWh = round(dispatchable_MWh);
452
453     if (this->dispatch_MWh != this->dispatchable_MWh) {
454         this->dispatch_MWh = this->dispatchable_MWh;
455     }
456
457     return;
458 }   /* __computeDispatch() */
```

4.14.3.4 __computeProduction()

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

Helper method to compute production values.

```
354 {
355     if (this->is_broken) {
```

```
356         this->production_MWh = 0;
357         return;
358     }
359
360     double production_MWh = 0;
361
362     for (int i = 0; i < 30; i++) {
363         this->production_vec_MWh[i] =
364             this->max_daily_production_MWh * this->capacity_factor_vec[i];
365
366         production_MWh += this->production_vec_MWh[i];
367     }
368
369     this->production_MWh = round(production_MWh);
370
371     return;
372 }   /* __computeProduction() */
```

4.14.3.5 __computeProductionCosts()

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

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

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

4.14.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
114 {
115     //  1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
127         this->tile_improvement_sprite_animated[i].setRotation(0);
128
129         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
130
131         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
132         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
133         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
134         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
135     }
136
137     //  2. draw production text
138     std::string production_string = "[W]:  INCREASE DISPATCH\n";
139     production_string            += "[S]:  DECREASE DISPATCH\n";
140     production_string            += "                       \n";
141
142     production_string            += "DISPATCH:  ";
143     production_string            += std::to_string(this->dispatch_MWh);
144     production_string            += " MWh (MAX ";
```

```
145     production_string            += std::to_string(this->dispatchable_MWh);
146     production_string            += ")\n";
147
148     production_string            += "O&M COST:  ";
149     production_string            += std::to_string(this->operation_maintenance_cost);
150     production_string            += " K\n";
151
152     sf::Text production_text(
153         production_string,
154         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
155         16
156     );
157
158     production_text.setOrigin(production_text.getLocalBounds().width / 2,0);
159     production_text.setFillColor(MONOCHROME_TEXT_GREEN);
160
161     production_text.setPosition(400 + 30, 400 - 45);
162
163     this->render_window_ptr->draw(production_text);
164
165     return;
166 }   /* __drawProductionMenu() */
```

4.14.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
598 {
599     //  1. draw power capacity upgrade sprite
600     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
601         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
602         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 32 - 20);
603
604         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
605         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
606
607         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
608         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
609
610         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
611
612         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
613         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
614         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
615     }
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(WAVE_ENERGY_CONVERTER_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() */
```

4.14.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
473 {
474     if (this->just_built) {
475         return;
476     }
477
478     switch (this->event_ptr->key.code) {
479         case (sf::Keyboard::U): {
480             this->__openUpgradeMenu();
481
482             break;
483         }
484
485
486         case (sf::Keyboard::W): {
487             if (this->production_menu_open) {
488                 this->dispatch_MWh++;
489
490                 if (this->dispatch_MWh > this->dispatchable_MWh) {
491                     this->dispatch_MWh = 0;
492                 }
```

```
493
494                 this->__computeProductionCosts();
495                 this->assets_manager_ptr->getSound("interface click")->play();
496             }
497
498             else if (this->upgrade_menu_open) {
499                 this->__upgradePowerCapacity();
500             }
501
502             break;
503         }
504
505
506         case (sf::Keyboard::S): {
507             if (this->production_menu_open) {
508                 this->dispatch_MWh--;
509
510                 if (this->dispatch_MWh < 0) {
511                     this->dispatch_MWh = this->dispatchable_MWh;
512                 }
513
514                 this->__computeProductionCosts();
515                 this->assets_manager_ptr->getSound("interface click")->play();
516             }
517
518             break;
519         }
520
521
522         case (sf::Keyboard::D): {
523             if (this->upgrade_menu_open) {
524                 this->__upgradeStorageCapacity();
525                 this->__computeProduction();
526                 this->__computeDispatch();
527             }
528
529             break;
530         }
531
532
533         default: {
534             // do nothing!
535
536             break;
537         }
538     }
539
540     return;
541 }   /* __handleKeyPressEvents() */
```

4.14.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
556 {
557     if (this->just_built) {
558         return;
559     }
560     switch (this->event_ptr->mouseButton.button) {
561         case (sf::Mouse::Left): {
562             //...
563
564             break;
565         }
566
567
568         case (sf::Mouse::Right): {
569             //...
570
571             break;
572         }
573
574
575         default: {
576             // do nothing!
577
```

```
578             break;
579         }
580     }
581
582     return;
583 }   /* __handleMouseButtonEvents() */
```

4.14.3.10 __repair()

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

Helper method to repair the wave energy converter.

Reimplemented from TileImprovement.

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

4.14.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
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() */
```

4.14.3.12 __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.

```
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() */
```

4.14.3.15 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.

```
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. draw first element of animated sprite
1108    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1109
1110
1111    //  4. draw second element of animated sprite
1112    if (this->is_running) {
1113        this->tile_improvement_sprite_animated[0].setPosition(
1114            this->position_x,
1115            this->position_y + this->bobbing_y * cos(
1116                (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1117            )
1118        );
1119
1120        this->tile_improvement_sprite_animated[1].setPosition(
1121            this->position_x,
1122            this->position_y + 1.25 * this->bobbing_y * sin(
1123                (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1124            )
1125        );
1126    }
1127
1128    else {
1129        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1130            this->tile_improvement_sprite_animated[i].setPosition(
1131                this->position_x,
1132                this->position_y + this->bobbing_y * cos(
1133                    (double)(0.4 * M_PI * this->frame) / FRAMES_PER_SECOND
1134                )
1135            );
1136        }
1137    }
1138
1139    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1140
1141
1142    //  5. draw storage upgrades
1143    for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1144        this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1145    }
1146
1147
1148    //  6. handle dispatch illustration
1149    if (this->dispatch_MWh > 0) {
1150        this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1151        this->__drawDispatch();
1152    }
1153
1154
1155    //  7. draw production menu
1156    if (this->production_menu_open) {
1157        this->render_window_ptr->draw(this->production_menu_backing);
1158        this->render_window_ptr->draw(this->production_menu_backing_text);
1159
1160        this->__drawProductionMenu();
1161    }
1162
```

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

```
844 {
845     //                  32 char x 17 line console "--------------------------------\n";
846     std::string options_substring                = "CAPACITY:      ";
847     options_substring                           += std::to_string(this->capacity_kW);
848     options_substring                           += " kW (level ";
849     options_substring                           += std::to_string(this->upgrade_level);
850     options_substring                           += ")\n";
851
852     options_substring                           += "PRODUCTION:    ";
853     options_substring                           += std::to_string(this->production_MWh);
854     options_substring                           += " MWh\n";
855
856     options_substring                           += "DISPATCHABLE:  ";
857     options_substring                           += std::to_string(this->dispatchable_MWh);
858     options_substring                           += " MWh\n";
859
860     options_substring                           += "HEALTH:      ";
861     options_substring                           += std::to_string(this->health);
862     options_substring                           += "/100";
863
864     if (this->health <= 0) {
865         options_substring                       += " ** BROKEN! **\n";
866     }
867
868     else {
869         options_substring                       += "\n";
870     }
871
872     options_substring                           += "                                \n";
873     options_substring                           += " **** WAVE ENERGY OPTIONS ****  \n";
874     options_substring                           += "                                \n";
875
876     if (this->is_broken) {
877         options_substring                       += "     [R]:  REPAIR (";
```

```
878         options_substring                       += std::to_string(WAVE_ENERGY_CONVERTER_BUILD_COST);
879         options_substring                       += " K)\n";
880     }
881
882     else {
883         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
884     }
885
886     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
887     options_substring                           += "HOLD [P]:  SCRAP (";
888     options_substring                           += std::to_string(SCRAP_COST);
889     options_substring                           += " K)";
890
891     return options_substring;
892 }   /* getTileOptionsSubstring() */
```

4.14.3.17 processEvent()

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

Method to process WaveEnergyConverter. 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() */
```

4.14.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

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

```
909 {
910     TileImprovement :: setIsSelected(is_selected);
911 
912     if (this->is_running and this->is_selected) {
913         this->assets_manager_ptr->getSound("ocean waves")->play();
914     }
915 
916     return;
917 }   /* setIsSelected() */
```

4.14.3.20 update()

```
void WaveEnergyConverter::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() */
```

4.14.4 Member Data Documentation

4.14.4.1 bobbing_y

```
double WaveEnergyConverter::bobbing_y
```

The bobbing extent of the wave energy converter.

4.14.4.2 capacity_factor_vec

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

A vector of daily capacity factors for the current month.

4.14.4.3 capacity_kW

```
int WaveEnergyConverter::capacity_kW
```

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

4.14.4.4 dispatch_MWh

```
int WaveEnergyConverter::dispatch_MWh
```

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

4.14.4.5 dispatch_vec_MWh

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

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

4.14.4.6 dispatchable_MWh

```
int WaveEnergyConverter::dispatchable_MWh
```

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

4.14.4.7 max_daily_production_MWh

```
double WaveEnergyConverter::max_daily_production_MWh
```

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

4.14.4.8 production_MWh

```
int WaveEnergyConverter::production_MWh
```

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

4.14.4.9 production_vec_MWh

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

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

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

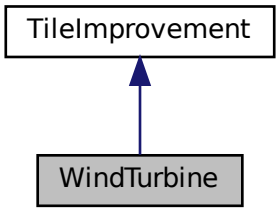
- header/WaveEnergyConverter.h
- source/WaveEnergyConverter.cpp

4.15 WindTurbine Class Reference

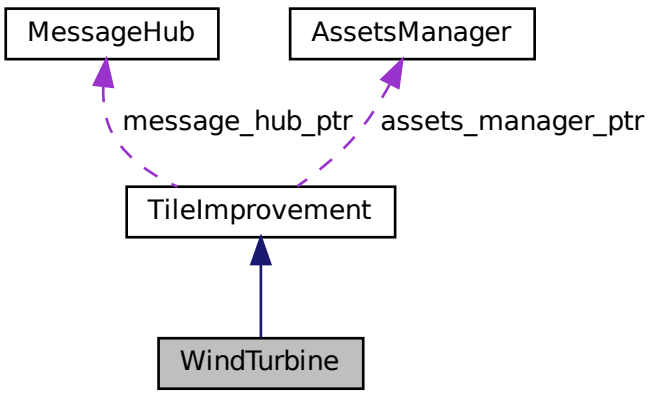
A settlement class (child class of TileImprovement).

```
#include <WindTurbine.h>
```

Inheritance diagram for WindTurbine:



Collaboration diagram for WindTurbine:



Public Member Functions

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

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

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

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

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

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

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

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

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

  Destructor for the WindTurbine class.

Public Attributes

- int capacity_kW

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

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

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

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

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

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

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

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

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

Private Member Functions

- void __setUpTileImprovementSpriteAnimated (void)

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

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

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

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

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

  Helper method to repair the wind turbine.
- void __computeCapacityFactors (void)

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

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

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

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

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

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

  Helper method to format and sent improvement state message.

Additional Inherited Members

4.15.1 Detailed Description

A settlement class (child class of TileImprovement).

4.15.2 Constructor & Destructor Documentation

4.15.2.1 WindTurbine()

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

Constructor for the WindTurbine class.

Ref: Wikipedia [2023]

Parameters

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

```
782   :
783 TileImprovement(
784     position_x,
785     position_y,
786     tile_resource,
787     event_ptr,
788     render_window_ptr,
789     assets_manager_ptr,
790     message_hub_ptr
791 )
792 {
793     //  1. set attributes
794
795     //  1.1. private
796     //...
797
798     //  1.2. public
799     this->tile_improvement_type = TileImprovementType :: WIND_TURBINE;
800
801     this->is_running = false;
802
803     this->health = 100;
804
805     this->capacity_kW = 100;
806     this->upgrade_level = 1;
807
808     this->storage_kWh = 0;
809     this->storage_level = 0;
810
811     this->production_MWh = 0;
812     this->dispatch_MWh = 0;
813     this->dispatchable_MWh = 0;
814
815     this->max_daily_production_MWh = (double)(24 * this->capacity_kW) / 1000;
816
817     this->rotor_drotation = 256 * SECONDS_PER_FRAME;
818
819     this->capacity_factor_vec.resize(30, 0);
820     this->production_vec_MWh.resize(30, 0);
821     this->dispatch_vec_MWh.resize(30, 0);
822
823     this->tile_improvement_string = "WIND TURBINE";
824
825     this->__setUpTileImprovementSpriteAnimated();
826     this->__computeCapacityFactors();
827     this->update();
828
829     std::cout « "WindTurbine constructed at " « this « std::endl;
830
831     return;
832 }   /* WindTurbine() */
```

4.15.2.2 ~WindTurbine()

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

Destructor for the WindTurbine class.

```
1188 {
1189     std::cout « "WindTurbine at " « this « " destroyed" « std::endl;
1190
1191     return;
1192 }   /* ~WindTurbine() */
```

4.15.3 Member Function Documentation

4.15.3.1 __breakdown()

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

Helper method to trigger an equipment breakdown.

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

4.15.3.2 __computeCapacityFactors()

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

Helper method to compute capacity factors.

```
307 {
308     if (this->is_broken) {
309         return;
310     }
311 
312     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
313     std::default_random_engine generator(seed);
314 
315     double mean =
316         this->tile_resource_scalar * MEAN_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
317 
318     double stdev = STDEV_DAILY_WIND_CAPACITY_FACTORS[this->month - 1];
319 
320     if (this->tile_resource_scalar > 1) {
321         stdev /= this->tile_resource_scalar;
322     }
323 
324     std::normal_distribution<double> normal_dist(mean, stdev);
325 
326     double capacity_factor = 0;
327 
328     for (int i = 0; i < 30; i++) {
329         capacity_factor = normal_dist(generator);
330 
331         if (capacity_factor < 0) {
332             capacity_factor = 0;
333         }
334 
335         this->capacity_factor_vec[i] = capacity_factor;
336     }
337 
338     return;
339 }   /* __computeCapacityFactors() */
```

4.15.3.3 __computeDispatch()

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

Helper method to compute dispatch values.

```
387 {
388     if (this->is_broken) {
389         this->dispatchable_MWh = 0;
390         return;
391     }
392
393     double stored_energy_MWh = 0;
394     double storage_capacity_MWh = (double)(this->storage_kWh) / 1000;
395
396     double demand_MWh = 0;
397     double production_MWh = 0;
398     double dispatchable_MWh = 0;
399     double difference_MWh = 0;
400
401     double room_MWh = 0;
402
403     for (int i = 0; i < 30; i++) {
404         demand_MWh = this->demand_vec_MWh[i];
405         production_MWh = this->production_vec_MWh[i];
406
407         if (production_MWh <= demand_MWh) {
408             this->dispatch_vec_MWh[i] = production_MWh;
409             dispatchable_MWh += this->dispatch_vec_MWh[i];
410
411             difference_MWh = demand_MWh - production_MWh;
412
413             if ((storage_capacity_MWh > 0) and (stored_energy_MWh > 0)) {
414                 if (difference_MWh > stored_energy_MWh) {
415                     this->dispatch_vec_MWh[i] += stored_energy_MWh;
416                     dispatchable_MWh += stored_energy_MWh;
417                     stored_energy_MWh = 0;
418                 }
419
420                 else {
421                     this->dispatch_vec_MWh[i] += difference_MWh;
422                     dispatchable_MWh += difference_MWh;
423                     stored_energy_MWh -= difference_MWh;
424                 }
425             }
426         }
427
428         else {
429             this->dispatch_vec_MWh[i] = demand_MWh;
430             dispatchable_MWh += this->dispatch_vec_MWh[i];
431
432             difference_MWh = production_MWh - demand_MWh;
433
434             if (
435                 (storage_capacity_MWh > 0) and
436                 (stored_energy_MWh < storage_capacity_MWh)
437             ) {
438                 room_MWh = storage_capacity_MWh - stored_energy_MWh;
439
440                 if (difference_MWh > room_MWh) {
441                     stored_energy_MWh += room_MWh;
442                 }
443
444                 else {
445                     stored_energy_MWh += difference_MWh;
446                 }
447             }
448         }
449     }
450
451     this->dispatchable_MWh = round(dispatchable_MWh);
452
453     if (this->dispatch_MWh != this->dispatchable_MWh) {
454         this->dispatch_MWh = this->dispatchable_MWh;
455     }
456
457     return;
458 }   /* __computeDispatch() */
```

4.15.3.4 __computeProduction()

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

Helper method to compute production values.

```
354 {
355     if (this->is_broken) {
356         this->production_MWh = 0;
357         return;
358     }
359
360     double production_MWh = 0;
361
362     for (int i = 0; i < 30; i++) {
363         this->production_vec_MWh[i] =
364             this->max_daily_production_MWh * this->capacity_factor_vec[i];
365
366         production_MWh += this->production_vec_MWh[i];
367     }
368
369     this->production_MWh = round(production_MWh);
370
371     return;
372 }   /* __computeProduction() */
```

4.15.3.5 __computeProductionCosts()

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

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

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

4.15.3.6 __drawProductionMenu()

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

Helper method to draw production menu assets.

```
114 {
115     //  1. draw static sprite
116     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
117         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
118         this->tile_improvement_sprite_animated[i].setPosition(400 - 138, 400 + 16);
119
120         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
121         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
122
123         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
124         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
125
126         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
127         this->tile_improvement_sprite_animated[i].setRotation(0);
128
129         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
130
131         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
132         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
133         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
```

```
134         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
135     }
136 
137     //  2. draw production text
138     std::string production_string = "[W]:  INCREASE DISPATCH\n";
139     production_string            += "[S]:  DECREASE DISPATCH\n";
140     production_string            += "                       \n";
141 
142     production_string            += "DISPATCH:  ";
143     production_string            += std::to_string(this->dispatch_MWh);
144     production_string            += " MWh (MAX ";
145     production_string            += std::to_string(this->dispatchable_MWh);
146     production_string            += ")\n";
147 
148     production_string            += "O&M COST:  ";
149     production_string            += std::to_string(this->operation_maintenance_cost);
150     production_string            += " K\n";
151 
152     sf::Text production_text(
153         production_string,
154         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
155         16
156     );
157 
158     production_text.setOrigin(production_text.getLocalBounds().width / 2,0);
159     production_text.setFillColor(MONOCHROME_TEXT_GREEN);
160 
161     production_text.setPosition(400 + 30, 400 - 45);
162 
163     this->render_window_ptr->draw(production_text);
164 
165     return;
166 }   /* __drawProductionMenu() */
```

4.15.3.7 __drawUpgradeOptions()

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

Helper method to set up and draw upgrade options.

```
599 {
600     //  1. draw power capacity upgrade sprite
601     for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
602         sf::Vector2f initial_position = this->tile_improvement_sprite_animated[i].getPosition();
603         this->tile_improvement_sprite_animated[i].setPosition(400 - 100, 400 - 56);
604 
605         sf::Color initial_colour = this->tile_improvement_sprite_animated[i].getColor();
606         this->tile_improvement_sprite_animated[i].setColor(sf::Color(255, 255, 255, 255));
607 
608         sf::Vector2f initial_scale = this->tile_improvement_sprite_animated[i].getScale();
609         this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1, 1));
610 
611         double initial_rotation = this->tile_improvement_sprite_animated[i].getRotation();
612         this->tile_improvement_sprite_animated[i].setRotation(0);
613 
614         this->render_window_ptr->draw(this->tile_improvement_sprite_animated[i]);
615 
616         this->tile_improvement_sprite_animated[i].setPosition(initial_position);
617         this->tile_improvement_sprite_animated[i].setColor(initial_colour);
618         this->tile_improvement_sprite_animated[i].setScale(initial_scale);
619         this->tile_improvement_sprite_animated[i].setRotation(initial_rotation);
620     }
621 
622     this->render_window_ptr->draw(this->upgrade_arrow_sprite);
623 
624 
625     //  2. draw power capacity upgrade text
626     //                  16 char line = "               \n"
627     std::string power_upgrade_string = "POWER CAPACITY  \n";
628     power_upgrade_string            += "               \n";
629 
630     power_upgrade_string            += "CAPACITY:  ";
631     power_upgrade_string            += std::to_string(this->capacity_kW);
632     power_upgrade_string            += " kW\n";
633 
634     power_upgrade_string            += "LEVEL:     ";
635     power_upgrade_string            += std::to_string(this->upgrade_level);
636     power_upgrade_string            += "\n\n";
```

```
637
638     if (this->upgrade_level < MAX_UPGRADE_LEVELS) {
639         power_upgrade_string        += "[W]: + 100 kW (";
640         power_upgrade_string        +=  std::to_string(WIND_TURBINE_BUILD_COST);
641         power_upgrade_string        += " K)\n";
642     }
643
644     else {
645         power_upgrade_string        += " * MAX LEVEL *  \n";
646     }
647
648     sf::Text power_upgrade_text = sf::Text(
649         power_upgrade_string,
650         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
651         16
652     );
653
654     power_upgrade_text.setOrigin(power_upgrade_text.getLocalBounds().width / 2, 0);
655     power_upgrade_text.setPosition(400 - 100, 400 - 32 + 16);
656     power_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
657
658     this->render_window_ptr->draw(power_upgrade_text);
659
660
661     //  3. draw energy capacity (storage) upgrade sprite
662     this->render_window_ptr->draw(this->storage_upgrade_sprite);
663     this->render_window_ptr->draw(this->upgrade_plus_sprite);
664
665
666     //  4. draw energy capacity (storage) upgrade text
667     //                   16 char line = "                \n"
668     std::string energy_upgrade_string = "ENERGY CAPACITY \n";
669     energy_upgrade_string             += "                \n";
670
671     energy_upgrade_string             += "CAPACITY:  ";
672     energy_upgrade_string             += std::to_string(this->storage_level * 200);
673     energy_upgrade_string             += " kWh\n";
674
675     energy_upgrade_string             += "LEVEL:     ";
676     energy_upgrade_string             += std::to_string(this->storage_level);
677     energy_upgrade_string             += "\n\n";
678
679     if (this->storage_level < MAX_STORAGE_LEVELS) {
680         energy_upgrade_string        += "[D]: + 200 kWh (";
681         energy_upgrade_string        +=  std::to_string(ENERGY_STORAGE_SYSTEM_BUILD_COST);
682         energy_upgrade_string        += " K)\n";
683     }
684
685     else {
686         energy_upgrade_string += " * MAX LEVEL *  \n";
687     }
688
689     sf::Text energy_upgrade_text = sf::Text(
690         energy_upgrade_string,
691         *(this->assets_manager_ptr->getFont("Glass_TTY_VT220")),
692         16
693     );
694
695     energy_upgrade_text.setOrigin(energy_upgrade_text.getLocalBounds().width / 2, 0);
696     energy_upgrade_text.setPosition(400 + 100, 400 - 32 + 16);
697     energy_upgrade_text.setFillColor(MONOCHROME_TEXT_GREEN);
698
699     this->render_window_ptr->draw(energy_upgrade_text);
700
701     return;
702 }   /* __drawUpgradeOptions() */
```

4.15.3.8 __handleKeyPressEvents()

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

Helper method to handle key press events.

```
473 {
474     if (this->just_built) {
475         return;
476     }
477
```

```
478     switch (this->event_ptr->key.code) {
479         case (sf::Keyboard::U): {
480             this->__openUpgradeMenu();
481 
482             break;
483         }
484 
485 
486         case (sf::Keyboard::W): {
487             if (this->production_menu_open) {
488                 this->dispatch_MWh++;
489 
490                 if (this->dispatch_MWh > this->dispatchable_MWh) {
491                     this->dispatch_MWh = 0;
492                 }
493 
494                 this->__computeProductionCosts();
495                 this->assets_manager_ptr->getSound("interface click")->play();
496             }
497 
498             else if (this->upgrade_menu_open) {
499                 this->__upgradePowerCapacity();
500             }
501 
502             break;
503         }
504 
505 
506         case (sf::Keyboard::S): {
507             if (this->production_menu_open) {
508                 this->dispatch_MWh--;
509 
510                 if (this->dispatch_MWh < 0) {
511                     this->dispatch_MWh = this->dispatchable_MWh;
512                 }
513 
514                 this->__computeProductionCosts();
515                 this->assets_manager_ptr->getSound("interface click")->play();
516             }
517 
518             break;
519         }
520 
521 
522         case (sf::Keyboard::D): {
523             if (this->upgrade_menu_open) {
524                 this->__upgradeStorageCapacity();
525                 this->__computeProduction();
526                 this->__computeDispatch();
527             }
528 
529             break;
530         }
531 
532 
533         default: {
534             // do nothing!
535 
536             break;
537         }
538     }
539 
540     return;
541 }   /* __handleKeyPressEvents() */
```

4.15.3.9 __handleMouseButtonEvents()

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

Helper method to handle mouse button events.

```
556 {
557     if (this->just_built) {
558         return;
559     }
560 
561     switch (this->event_ptr->mouseButton.button) {
562         case (sf::Mouse::Left): {
```

```
563             //...
564
565             break;
566         }
567
568
569         case (sf::Mouse::Right): {
570             //...
571
572             break;
573         }
574
575
576         default: {
577             // do nothing!
578
579             break;
580         }
581     }
582
583     return;
584 }   /* __handleMouseButtonEvents() */
```

4.15.3.10 __repair()

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

Helper method to repair the wind turbine.

Reimplemented from TileImprovement.

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

4.15.3.11 __sendImprovementStateMessage()

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

Helper method to format and sent improvement state message.

```
717 {
718     Message improvement_state_message;
719
720     improvement_state_message.channel = GAME_CHANNEL;
721     improvement_state_message.subject = "improvement state";
722
723     improvement_state_message.int_payload["dispatch_MWh"] = this->dispatch_MWh;
724     improvement_state_message.int_payload["operation_maintenance_cost"] =
725         this->operation_maintenance_cost;
726
727     this->message_hub_ptr->sendMessage(improvement_state_message);
728
729     std::cout « "Improvement state message sent by " « this « std::endl;
730
731     return;
732 }   /* __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.

```
937 {
938     //  1. send improvement state message
939     this->__sendImprovementStateMessage();
940 
941     //  2. update
942     this->__computeCapacityFactors();
943     this->update();
944 
945     //  3. handle start/stop
946     if ((not this->is_running) and (this->dispatch_MWh > 0)) {
947         this->is_running = true;
948     }
949 
950     else if (this->is_running and (this->dispatch_MWh <= 0)) {
951         this->is_running = false;
952     }
953 
954     //  4. handle equipment health and breakdowns
955     if (this->is_running) {
956         this->health--;
957 
958         if (this->health <= 50) {
959             double breakdown_prob = (51 - this->health) * BREAKDOWN_PROBABILITY_INCREMENT;
960 
961             if ((double)rand() / RAND_MAX <= breakdown_prob) {
962                 this->health = 0;
963             }
964         }
965 
966         if (this->health <= 0) {
967             this->__breakdown();
968         }
969     }
970 
971     //  5. send tile state request (if selected)
972     if (this->is_selected) {
973         this->__sendTileStateRequest();
974     }
975 
976     return;
977 }   /* 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.

```
1068 {
1069     //  1. if just built, call base method and return
1070     if (this->just_built) {
1071         TileImprovement :: draw();
1072 
1073         return;
1074     }
1075 
1076 
1077     //  2. handle upgrade effects
1078     if (this->just_upgraded) {
1079         for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1080             this->tile_improvement_sprite_animated[i].setColor(
1081                 sf::Color(
1082                     255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
```

```
1083                    255,
1084                    255 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1085                    255
1086                )
1087            );
1088
1089            this->tile_improvement_sprite_animated[i].setScale(
1090                sf::Vector2f(
1091                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2),
1092                    1 + 0.2 * pow(cos((M_PI * this->upgrade_frame) / FRAMES_PER_SECOND), 2)
1093                )
1094            );
1095        }
1096
1097        this->upgrade_frame++;
1098    }
1099
1100    if (this->upgrade_frame >= 2 * FRAMES_PER_SECOND) {
1101        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1102            this->tile_improvement_sprite_animated[i].setColor(
1103                sf::Color(255,255,255,255)
1104            );
1105
1106            this->tile_improvement_sprite_animated[i].setScale(sf::Vector2f(1,1));
1107        }
1108
1109        this->just_upgraded = false;
1110        this->upgrade_frame = 0;
1111    }
1112
1113
1114    //  3. draw first element of animated sprite
1115    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[0]);
1116
1117
1118    //  4. draw second element of animated sprite
1119    if (this->is_running) {
1120        this->tile_improvement_sprite_animated[1].rotate(this->rotor_drotation);
1121    }
1122
1123    this->render_window_ptr->draw(this->tile_improvement_sprite_animated[1]);
1124
1125
1126    //  5. draw storage upgrades
1127    for (size_t i = 0; i < this->storage_upgrade_sprite_vec.size(); i++) {
1128        this->render_window_ptr->draw(this->storage_upgrade_sprite_vec[i]);
1129    }
1130
1131
1132    //  6. handle dispatch illustration
1133    if (this->dispatch_MWh > 0) {
1134        this->dispatch_text.setString(std::to_string(this->dispatch_MWh));
1135        this->__drawDispatch();
1136    }
1137
1138
1139    //  7. draw production menu
1140    if (this->production_menu_open) {
1141        this->render_window_ptr->draw(this->production_menu_backing);
1142        this->render_window_ptr->draw(this->production_menu_backing_text);
1143
1144        this->__drawProductionMenu();
1145    }
1146
1147
1148    //  8. draw upgrade menu
1149    if (this->upgrade_menu_open) {
1150        this->render_window_ptr->draw(this->upgrade_menu_backing);
1151        this->render_window_ptr->draw(this->upgrade_menu_backing_text);
1152
1153        this->__drawUpgradeOptions();
1154    }
1155
1156
1157    //  9. handle broken effects
1158    if (this->is_broken) {
1159        for (size_t i = 0; i < this->tile_improvement_sprite_animated.size(); i++) {
1160            this->tile_improvement_sprite_animated[i].setColor(
1161                sf::Color(
1162                    255,
1163                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1164                    255 * pow(cos((M_PI * this->frame) / FRAMES_PER_SECOND), 2),
1165                    255
1166                )
1167            );
1168        }
1169    }
```

```
1170
1171     this->frame++;
1172     return;
1173 }   /* 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.

```
849 {
850     //                  32 char x 17 line console "--------------------------------\n";
851     std::string options_substring                = "CAPACITY:      ";
852     options_substring                           += std::to_string(this->capacity_kW);
853     options_substring                           += " kW (level ";
854     options_substring                           += std::to_string(this->upgrade_level);
855     options_substring                           += ")\n";
856
857     options_substring                           += "PRODUCTION:    ";
858     options_substring                           += std::to_string(this->production_MWh);
859     options_substring                           += " MWh\n";
860
861     options_substring                           += "DISPATCHABLE:  ";
862     options_substring                           += std::to_string(this->dispatchable_MWh);
863     options_substring                           += " MWh\n";
864
865     options_substring                           += "HEALTH:      ";
866     options_substring                           += std::to_string(this->health);
867     options_substring                           += "/100";
868
869     if (this->health <= 0) {
870         options_substring                       += " ** BROKEN! **\n";
871     }
872
873     else {
874         options_substring                       += "\n";
875     }
876
877     options_substring                           += "                                \n";
878     options_substring                           += " **** WIND TURBINE OPTIONS **** \n";
879     options_substring                           += "                                \n";
880
881     if (this->is_broken) {
882         options_substring                       += "     [R]:  REPAIR (";
883         options_substring                       += std::to_string(WIND_TURBINE_BUILD_COST);
884         options_substring                       += " K)\n";
885     }
886
887     else {
888         options_substring                       += "     [E]:  OPEN PRODUCTION MENU \n";
889     }
890
891     options_substring                           += "     [U]:  OPEN UPGRADE MENU    \n";
892     options_substring                           += "HOLD [P]:  SCRAP (";
893     options_substring                           += std::to_string(SCRAP_COST);
894     options_substring                           += " K)";
895
896     return options_substring;
897 }   /* getTileOptionsSubstring() */
```

4.15.3.17 processEvent()

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

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

Reimplemented from TileImprovement.

```
1019 {
1020     TileImprovement :: processEvent();
1021
1022     if (this->event_ptr->type == sf::Event::KeyPressed) {
1023         this->__handleKeyPressEvents();
1024     }
1025
1026     if (this->event_ptr->type == sf::Event::MouseButtonPressed) {
1027         this->__handleMouseButtonEvents();
1028     }
1029
1030     return;
1031 }   /* processEvent() */
```

4.15.3.18 processMessage()

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

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

Reimplemented from TileImprovement.

```
1046 {
1047     TileImprovement :: processMessage();
1048
1049     //...
1050
1051     return;
1052 }   /* 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.

```
914 {
915     TileImprovement :: setIsSelected(is_selected);
916
917     if (this->is_running and this->is_selected) {
918         this->assets_manager_ptr->getSound("wind turbine running")->play();
919     }
920
```

```
921     return;
922 }   /* setIsSelected() */
```

4.15.3.20 update()

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

Method to trigger production and dispatchable updates.

Reimplemented from TileImprovement.

```
992 {
993     std::cout « "WindTurbine :: update()" « std::endl;
994
995     this->__computeProduction();
996     this->__computeProductionCosts();
997     this->__computeDispatch();
998
999     if (this->is_selected) {
1000         this->__sendTileStateRequest();
1001     }
1002
1003     return;
1004 }   /* 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

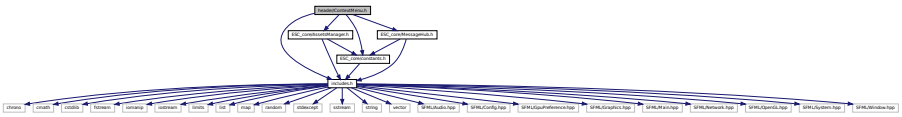
File Documentation

5.1 header/ContextMenu.h File Reference

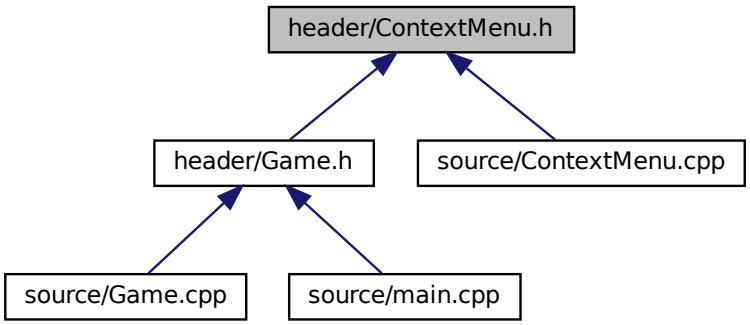
Header file for the ContextMenu class.

```
#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

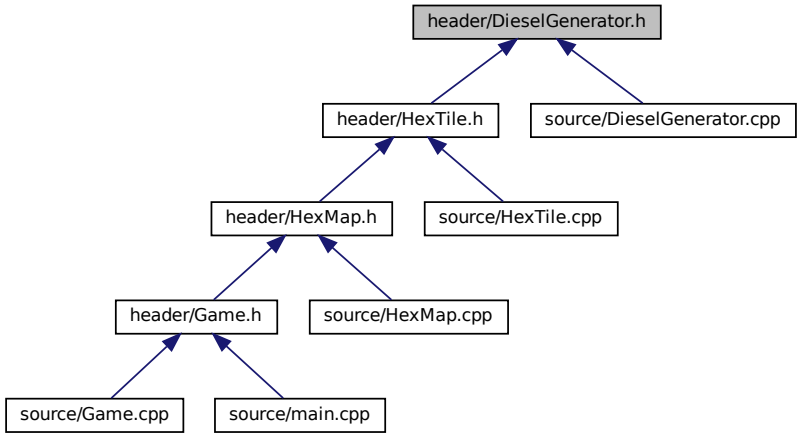
Header file for the DieselGenerator class.

```
#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.

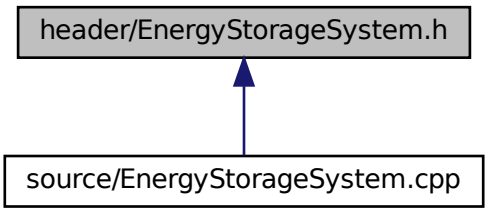
5.3 header/EnergyStorageSystem.h File Reference

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).

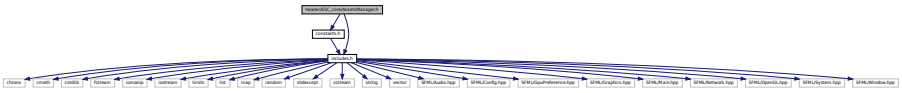
5.3.1 Detailed Description

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

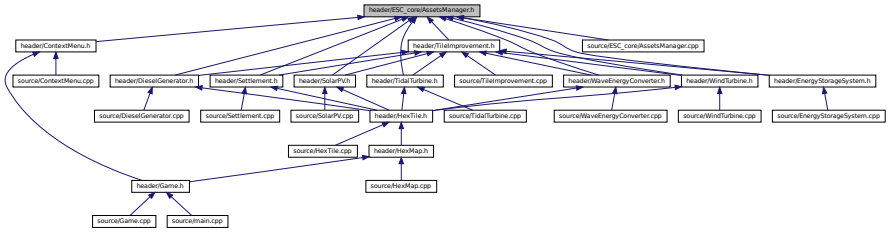
5.4 header/ESC_core/AssetsManager.h File Reference

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.

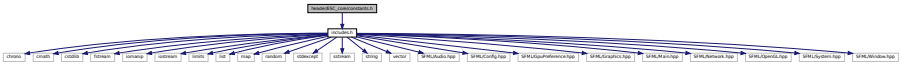
5.4.1 Detailed Description

Header file for the AssetsManager class.

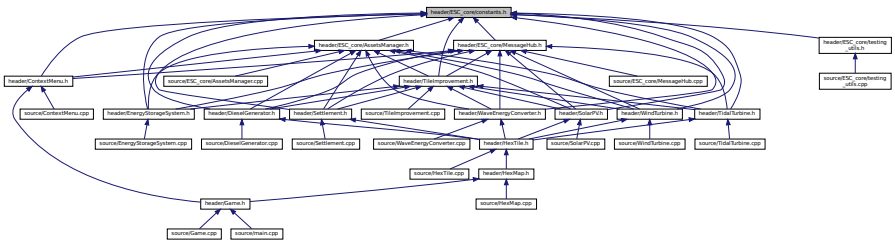
5.5 header/ESC_core/constants.h File Reference

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

  Target frames per second.
- const double SECONDS_PER_FRAME = 1.0 / 60

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

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

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

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

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

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

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

A message channel for tile state messages.

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

  A message channel for hex map messages.
- const std::string SETTLEMENT_CHANNEL = "SETTLEMENT CHANNEL"

  A message channel for the settlement.
- const int CLEAR_FOREST_COST = 160

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

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

  The cost of clearing a plains tile.
- const int DIESEL_GENERATOR_BUILD_COST = 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 int SCRAP_COST = 50

  The cost of scrapping a tile improvement (other than settlement).
- const int MAX_UPGRADE_LEVELS = 5

  The maximum upgrade level of any tile improvement.
- const int MAX_STORAGE_LEVELS = 5

  The maximum storage level of any tile improvement.
- const int STARTING_CREDITS = 800

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

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

  The lifetime limit on CO2-equivalent emissions (1 tonne CO2e ∼= 667 L diesel).
- const int RESOURCE_ASSESSMENT_COST = 20

  The cost of doing a resource assessment.
- const int BUILD_SETTLEMENT_COST = 250

  The cost of building a settlement.
- const int STARTING_POPULATION = 100

  The starting population of a settlement.
- const double MEAN_POPULATION_GROWTH_RATE = 0.020

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

  The standard deviation in monthly population growth rate.

- const double LITRES_DIESEL_PER_MWH_PRODUCTION = 375

  The litres of diesel consumed in producing 1 MWh (assumes higher heating value and constant thermal efficiency of ∼0.25).
- const double COST_PER_LITRE_DIESEL = 1.75

  The cost of a litre of diesel.
- const double KG_CO2E_PER_LITRE_DIESEL = 3.16

  The CO2-equivalent mass of emissions that result from burning one litre of diesel fuel.
- const double DIESEL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenace cost of running a diesel generator (assumed 0.05 credits per kWh produced).
- const double SOLAR_OP_MAINT_COST_PER_MWH_PRODUCTION = 10

  The operation and maintenance cost of running a solar PV array (assumed 0.01 credits per kWh produced).
- const double TIDAL_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a tidal turbine (assumed 0.05 credits per kWh produced).
- const double WAVE_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a wave energy converter (assumed 0.05 credits per kWh produced).
- const double WIND_OP_MAINT_COST_PER_MWH_PRODUCTION = 50

  The operation and maintenance cost of running a wind turbine (assumed 0.05 credits per kWh produced).
- const std::vector< double > MEAN_DAILY_DEMAND_RATIOS

  The mean daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_DEMAND_RATIOS

  The standard deviation in daily demand ratio for each month, where demand ratio is demand [MWh] divided by maximum daily demand [MWh]. Maximum daily demand is simply (24)(max load [kW]) / 1000.
- const double MAXIMUM_DAILY_DEMAND_PER_CAPITA = 0.05

  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 = 0.225

  The daily tidal capacity factor, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000. The tides are not a random process (usually semi-diurnal, mostly driven by orbits of moon and sun).
- const std::vector< double > MEAN_DAILY_WAVE_CAPACITY_FACTORS

  The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_WAVE_CAPACITY_FACTORS

  The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > MEAN_DAILY_WIND_CAPACITY_FACTORS

  The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::vector< double > STDEV_DAILY_WIND_CAPACITY_FACTORS

  The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.
- const std::string GAME_CHANNEL = "GAME CHANNEL"

  A message channel for game messages.
- const std::string GAME_STATE_CHANNEL = "GAME STATE CHANNEL"

  A message channel for game state messages.
- 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 = 1.125
```

The number of credits (x1000) earned.

5.5.3.8 DAILY_TIDAL_CAPACITY_FACTOR

```
const double DAILY_TIDAL_CAPACITY_FACTOR = 0.225
```

The daily tidal capacity factor, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000. The tides are not a random process (usually semi-diurnal, mostly driven by orbits of moon and sun).

5.5.3.9 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 = 2000
```

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_WIDTH

```
const int GAME_WIDTH = 1200
```

Width of the game space.

5.5.3.19 HEX_MAP_CHANNEL

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

A message channel for hex map messages.

5.5.3.20 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.

5.5.3.21 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).

5.5.3.22 MAX_STORAGE_LEVELS

```
const int MAX_STORAGE_LEVELS = 5
```

The maximum storage level of any tile improvement.

5.5.3.23 MAX_UPGRADE_LEVELS

```
const int MAX_UPGRADE_LEVELS = 5
```

The maximum upgrade level of any tile improvement.

5.5.3.24 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.

5.5.3.25 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.

5.5.3.26 MEAN_DAILY_SOLAR_CAPACITY_FACTORS

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

Initial value:

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

The mean daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.27 MEAN_DAILY_WAVE_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.742, 0.694, 0.618,
    0.467, 0.366, 0.292,
    0.280, 0.293, 0.374,
    0.424, 0.662, 0.600
}
```

The mean daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.28 MEAN_DAILY_WIND_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.591, 0.594, 0.627,
    0.629, 0.579, 0.537,
    0.442, 0.507, 0.587,
    0.618, 0.611, 0.580
}
```

The mean daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.29 MEAN_POPULATION_GROWTH_RATE

```
const double MEAN_POPULATION_GROWTH_RATE = 0.020
```

The mean monthly population growth rate.

5.5.3.30 NO_TILE_SELECTED_CHANNEL

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

A message channel for no tile selected messages.

5.5.3.31 RESOURCE_ASSESSMENT_COST

```
const int RESOURCE_ASSESSMENT_COST = 20
```

The cost of doing a resource assessment.

5.5.3.32 SCRAP_COST

```
const int SCRAP_COST = 50
```

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

5.5.3.33 SECONDS_PER_FRAME

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

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

5.5.3.34 SECONDS_PER_MONTH

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

5.5.3.35 SECONDS_PER_YEAR

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

5.5.3.36 SETTLEMENT_CHANNEL

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

A message channel for the settlement.

5.5.3.37 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).

5.5.3.38 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.

5.5.3.39 SOLAR_PV_WATER_BUILD_MULTIPLIER

```
const double SOLAR_PV_WATER_BUILD_MULTIPLIER = 1.285714
```

The additional cost of building on water.

5.5.3.40 STARTING_CREDITS

```
const int STARTING_CREDITS = 800
```

The starting balance of credits.

5.5.3.41 STARTING_POPULATION

```
const int STARTING_POPULATION = 100
```

The starting population of a settlement.

5.5.3.42 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.

5.5.3.43 STDEV_DAILY_SOLAR_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.013, 0.024, 0.043,
    0.049, 0.072, 0.072,
    0.076, 0.065, 0.048,
    0.026, 0.018, 0.009
}
```

The standard deviation in daily solar capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.44 STDEV_DAILY_WAVE_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.146, 0.135, 0.163,
    0.145, 0.158, 0.106,
    0.086, 0.058, 0.145,
    0.171, 0.184, 0.309
}
```

The standard deviation in daily wave capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.45 STDEV_DAILY_WIND_CAPACITY_FACTORS

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

Initial value:

```
= {
    0.147, 0.142, 0.198,
    0.154, 0.162, 0.202,
    0.180, 0.217, 0.198,
    0.168, 0.141, 0.168
}
```

The standard deviation in daily wind capacity factors for each month, where capacity factor is daily production [MWh] divided by maximum daily production [MWh]. Maximum daily production is simply (24)(power capacity [kW]) / 1000.

5.5.3.46 STDEV_POPULATION_GROWTH_RATE

```
const double STDEV_POPULATION_GROWTH_RATE = 0.005
```

The standard deviation in monthly population growth rate.

5.5.3.47 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).

5.5.3.48 TIDAL_TURBINE_BUILD_COST

```
const int TIDAL_TURBINE_BUILD_COST = 550
```

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

5.5.3.49 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).

5.5.3.50 TILE_SELECTED_CHANNEL

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

A message channel for tile selection messages.

5.5.3.51 TILE_STATE_CHANNEL

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

A message channel for tile state messages.

5.5.3.52 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).

5.5.3.53 TUTORIAL_PAGES

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

5.5.3.54 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.

5.5.3.55 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).

5.5.3.56 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).

5.5.3.57 WIND_TURBINE_BUILD_COST

```
const int WIND_TURBINE_BUILD_COST = 450
```

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

5.5.3.58 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.

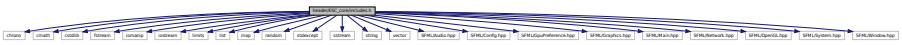
Ref: van Heesch. [2023]

5.7 header/ESC_core/includes.h File Reference

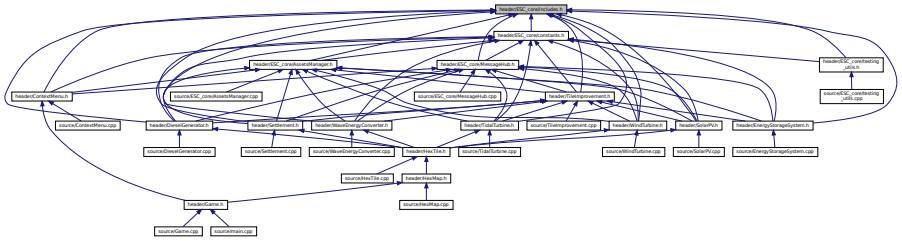
Header file for various includes.

```
#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.

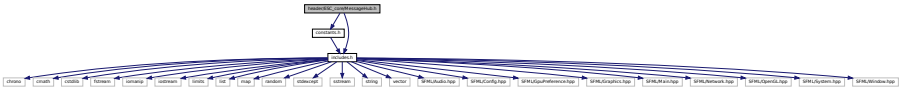
Ref: Gomila [2023]

5.8 header/ESC_core/MessageHub.h File Reference

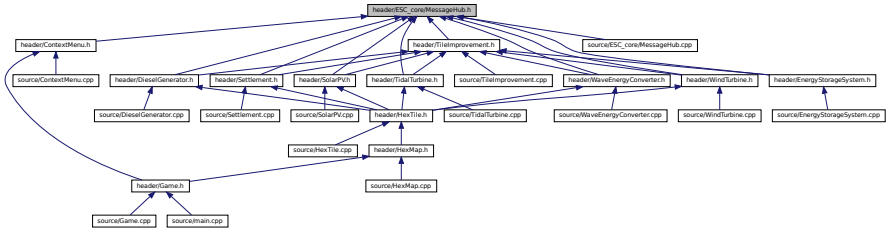
Header file for the MessageHub class.

```
#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

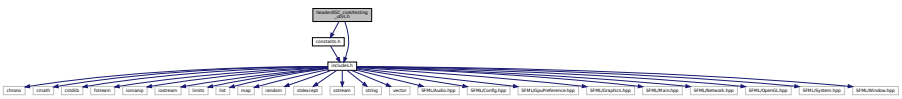
Header file for the MessageHub class.

5.9 header/ESC_core/testing_utils.h File Reference

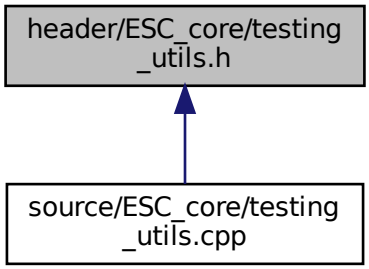
Header file for various testing utilities.

```
#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.

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__").

```
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.
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

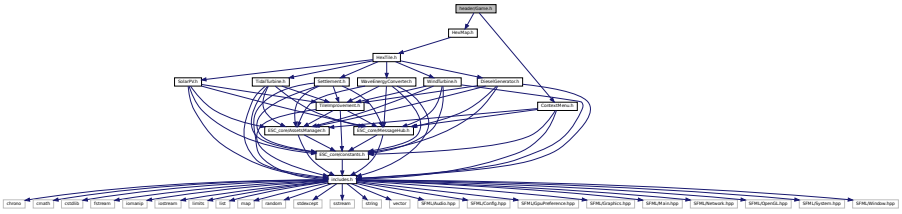
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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() */
```

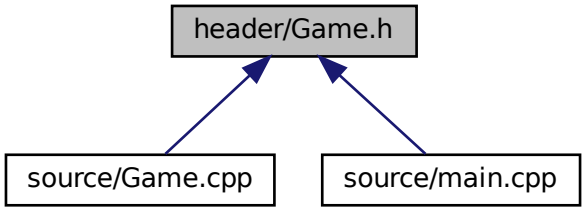
5.10 header/Game.h File Reference

```
#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.

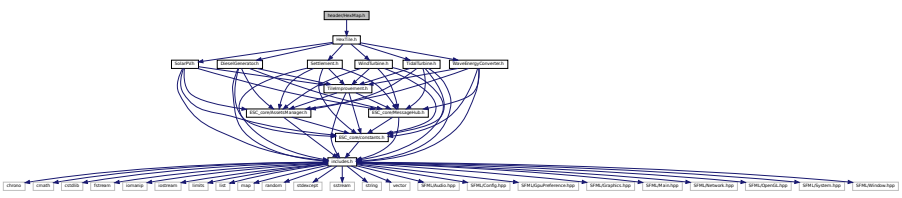
```
66                {
67     BUILD_SETTLEMENT,
68     SYSTEM_MANAGEMENT,
69     LOSS_EMISSIONS,
70     LOSS_DEMAND,
71     LOSS_CREDITS,
72     VICTORY,
73     N_GAME_PHASES
74 };  /* GamePhase */
```

5.11 header/HexMap.h File Reference

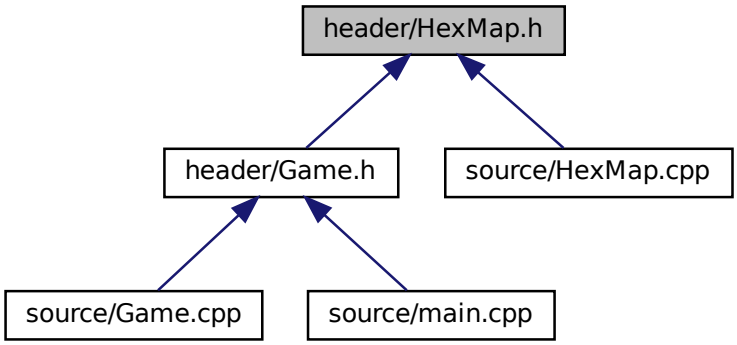
Header file for the HexMap class.

```
#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

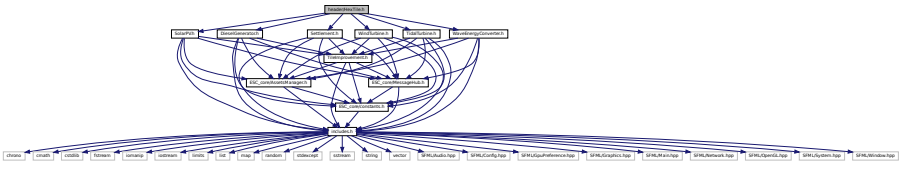
Header file for the HexMap class.

5.12 header/HexTile.h File Reference

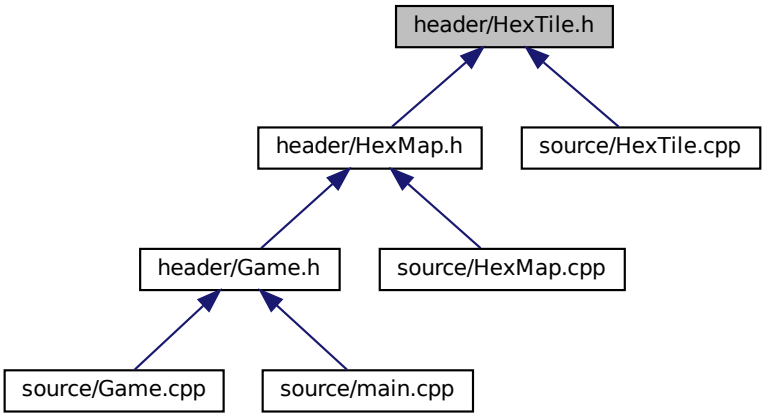
Header file for the Game class.

```
#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

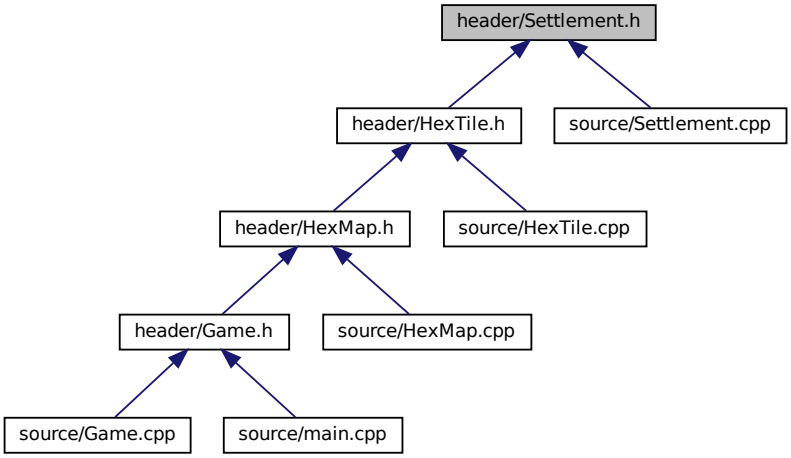
Header file for the Settlement class.

```
#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

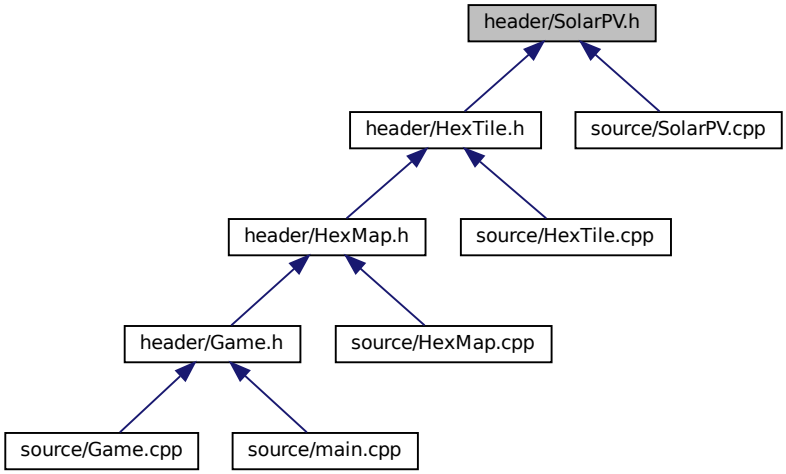
Header file for the SolarPV class.

```
#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

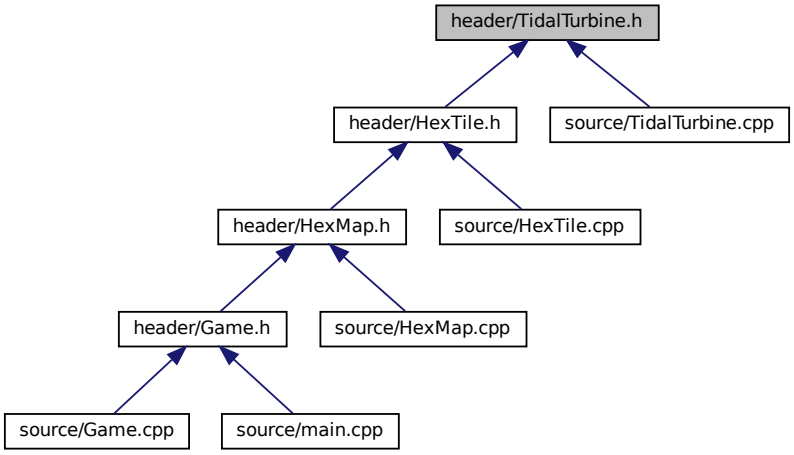
Header file for the TidalTurbine class.

```
#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.

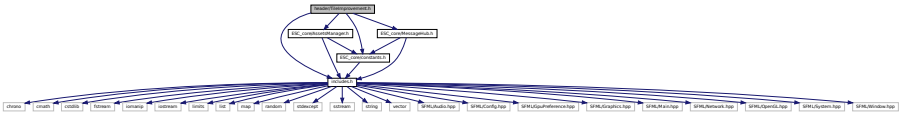
5.16 header/TileImprovement.h File Reference

Header file for the TileImprovement class.

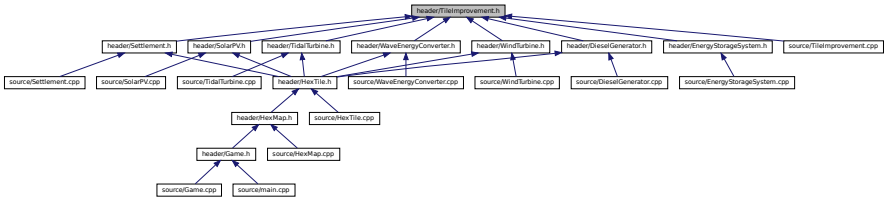
```
#include "ESC_core/constants.h"
#include "ESC_core/includes.h"
#include "ESC_core/AssetsManager.h"
```

```
#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.

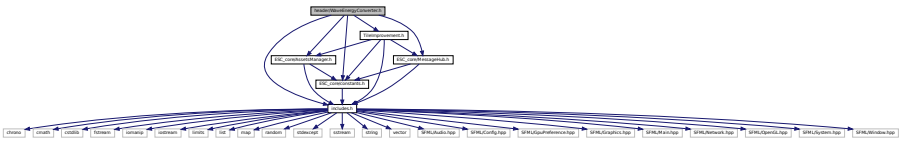
```
68                          {
69     SETTLEMENT,
70     DIESEL_GENERATOR,
71     SOLAR_PV,
72     WIND_TURBINE,
73     TIDAL_TURBINE,
74     WAVE_ENERGY_CONVERTER,
75     N_TILE_IMPROVEMENT_TYPES
76 };  /* TileImprovementType */
```

5.17 header/WaveEnergyConverter.h File Reference

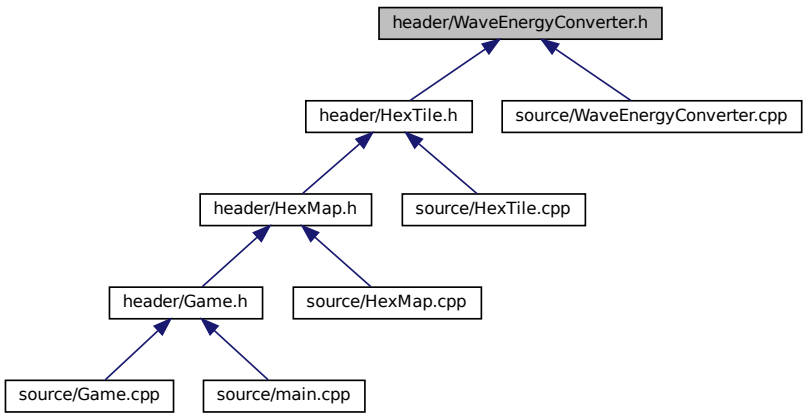
Header file for the WaveEnergyConverter class.

```
#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

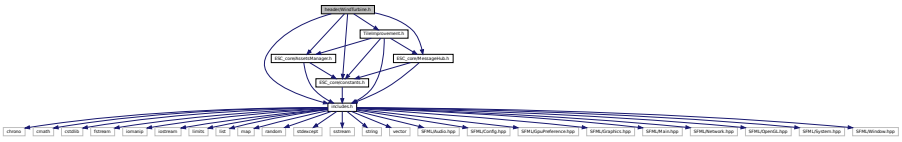
Header file for the WaveEnergyConverter class.

5.18 header/WindTurbine.h File Reference

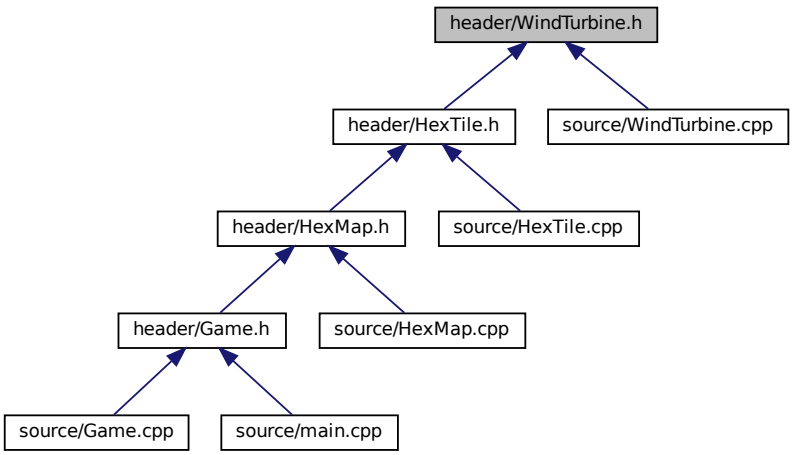
Header file for the WindTurbine class.

```
#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).

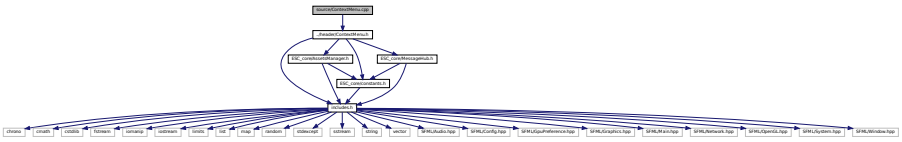
5.18.1 Detailed Description

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

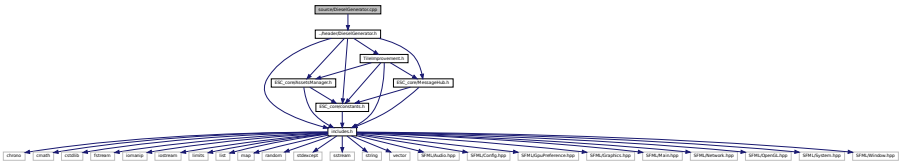
Implementation file for the ContextMenu class.

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

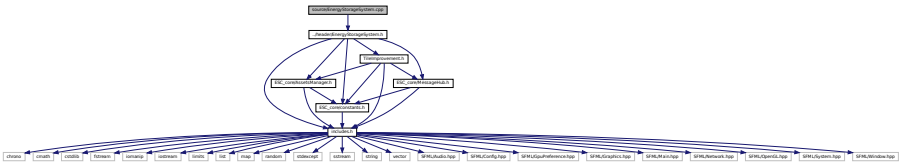
Implementation file for the DieselGenerator class.

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

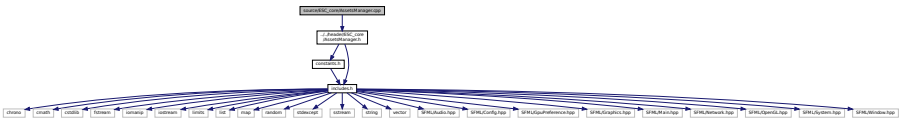
Implementation file for the EnergyStorageSystem class. DEPRECATED / NOT USED.

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

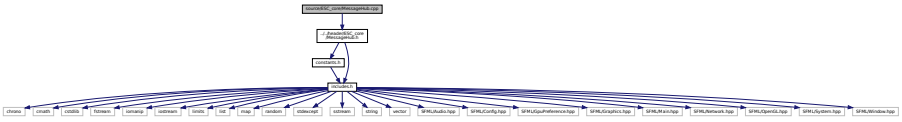
Implementation file for the AssetsManager class.

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.

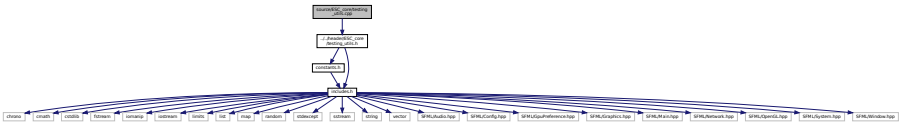
A class which acts as a central hub for inter-object message traffic.

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

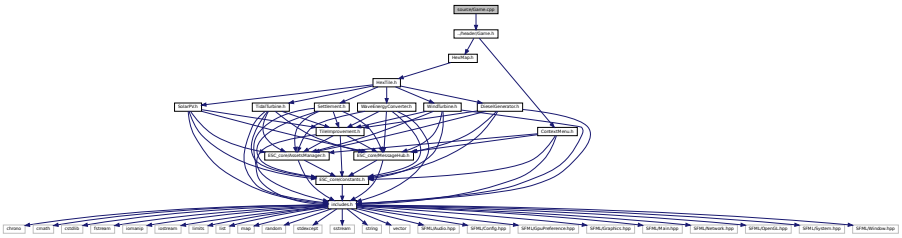
statement	The statement whose truth is to be tested ("1 == 0", for example).
file	The file in which the test is applied (you should be able to just pass in "__FILE__").
line	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
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

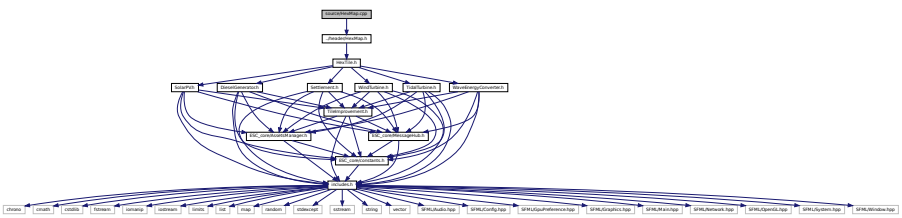
Implementation file for the Game class.

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

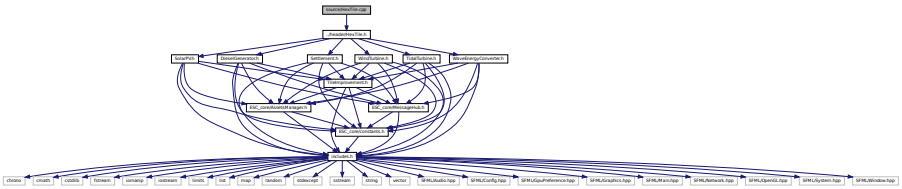
Implementation file for the HexMap class.

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

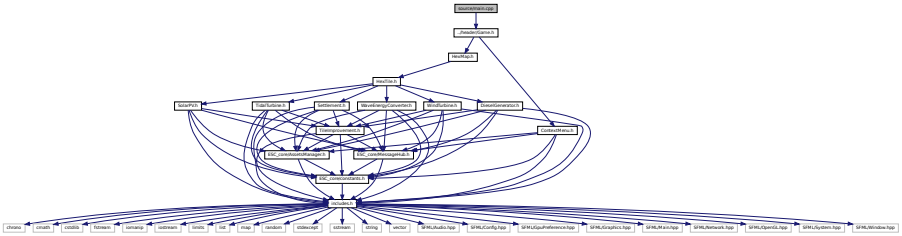
Implementation file for the HexTile class.

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.
- 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.

```
344 {
345     sf::RenderWindow* render_window_ptr = new sf::RenderWindow(
346         sf::VideoMode(GAME_WIDTH, GAME_HEIGHT),
347         "Road To Zero"
348     );
349 
350     return render_window_ptr;
351 }   /* 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()

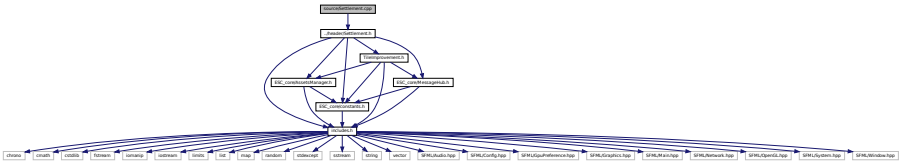
```
int main (
              int argc,
              char ** argv )
360 {
361     //  1. load assets
```

```
362     AssetsManager assets_manager;
363     loadAssets(&assets_manager);
364
365     //  2. construct render window
366     sf::RenderWindow* render_window_ptr = constructRenderWindow();
367
368     //  3. show brand animation and splash screen
369     //...
370
371     //  4. show game title
372     //...
373
374     //  5. start game loop
375     bool quit_game = false;
376     assets_manager.playTrack();
377
378     while (not quit_game) {
379         Game game(render_window_ptr, &assets_manager);
380         quit_game = game.run();
381     }
382
383     //  4. clean up
384     render_window_ptr->close();
385     delete render_window_ptr;
386
387     return 0;
388 }   /* main() */
```

5.29 source/Settlement.cpp File Reference

Implementation file for the Settlement class.

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



5.29.1 Detailed Description

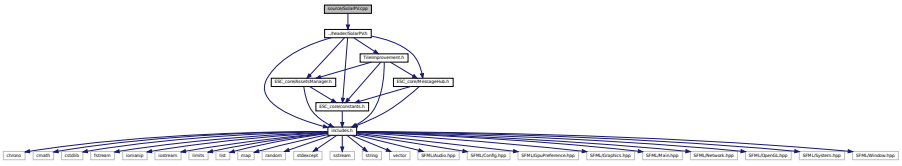
Implementation file for the Settlement class.

A base class for the tile improvement hierarchy.

5.30 source/SolarPV.cpp File Reference

Implementation file for the SolarPV class.

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



5.30.1 Detailed Description

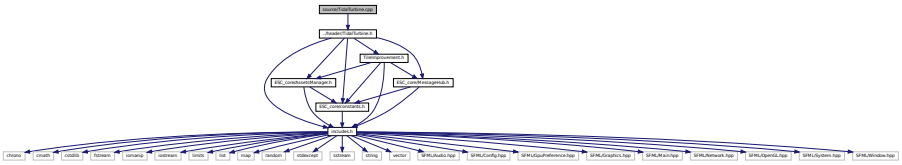
Implementation file for the SolarPV class.

A base class for the tile improvement hierarchy.

5.31 source/TidalTurbine.cpp File Reference

Implementation file for the TidalTurbine class.

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



5.31.1 Detailed Description

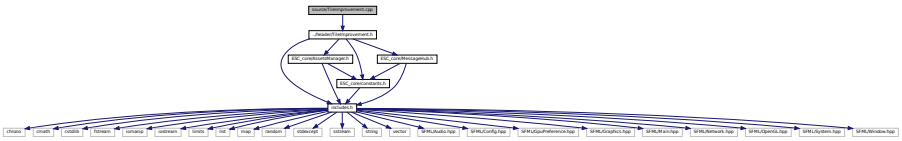
Implementation file for the TidalTurbine class.

A base class for the tile improvement hierarchy.

5.32 source/TileImprovement.cpp File Reference

Implementation file for the TileImprovement class.

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



5.32.1 Detailed Description

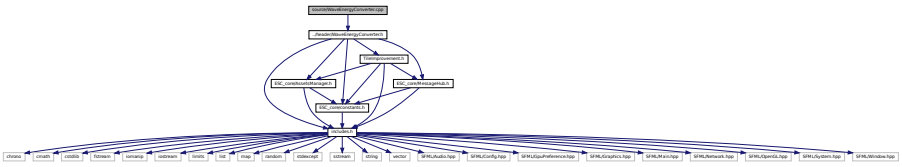
Implementation file for the TileImprovement class.

A base class for the tile improvement hierarchy.

5.33 source/WaveEnergyConverter.cpp File Reference

Implementation file for the WaveEnergyConverter class.

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



5.33.1 Detailed Description

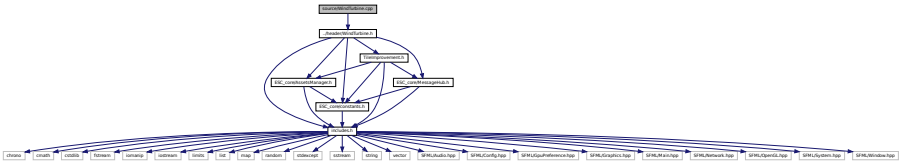
Implementation file for the WaveEnergyConverter class.

A base class for the tile improvement hierarchy.

5.34 source/WindTurbine.cpp File Reference

Implementation file for the WindTurbine class.

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



5.34.1 Detailed Description

Implementation file for the WindTurbine class.

A base class for the tile improvement hierarchy.

Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL https://www.sfml-dev.org/. 307

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL https://www.doxygen.nl. 306

Wikipedia. Hexagon, 2023. URL https://en.wikipedia.org/wiki/Hexagon. 40, 54, 128, 181, 191, 208, 228, 250, 268

Index