

HelloWorld

Generated by Doxygen 1.9.1



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 Class Documentation</b>	<b>5</b>
3.1 AssetsManager Class Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 AssetsManager()	6
3.1.2.2 ~AssetsManager()	6
3.1.3 Member Function Documentation	6
3.1.3.1 loadFont()	6
3.1.3.2 loadSound()	6
3.1.3.3 loadSoundBuffer()	7
3.1.3.4 loadTexture()	7
3.1.3.5 loadTrack()	7
3.1.4 Member Data Documentation	7
3.1.4.1 current_track	7
3.1.4.2 font_map	7
3.1.4.3 sound_map	7
3.1.4.4 soundbuffer_map	8
3.1.4.5 texture_map	8
3.1.4.6 track_map	8
3.2 InputsHandler Class Reference	8
3.2.1 Detailed Description	9
3.2.2 Constructor & Destructor Documentation	9
3.2.2.1 InputsHandler()	9
3.2.2.2 ~InputsHandler()	9
3.2.3 Member Function Documentation	9
3.2.3.1 __constructKeyCodeMap()	10
3.2.3.2 printKeysPressed()	13
3.2.3.3 process()	14
3.2.3.4 reset()	14
3.2.4 Member Data Documentation	15
3.2.4.1 key_code_map	15
3.2.4.2 key_press_vec	15
3.2.4.3 key_pressed_once_vec	15
<b>4 File Documentation</b>	<b>17</b>
4.1 header/ESC_core/AssetsManager.h File Reference	17
4.1.1 Detailed Description	17

4.2 header/ESC_core/constants.h File Reference	18
4.2.1 Detailed Description	18
4.2.2 Variable Documentation	18
4.2.2.1 FRAMES_PER_SECOND	18
4.2.2.2 SECONDS_PER_FRAME	18
4.3 header/ESC_core/doxygen_cite.h File Reference	18
4.3.1 Detailed Description	19
4.4 header/ESC_core/includes.h File Reference	19
4.4.1 Detailed Description	20
4.5 header/ESC_core/InputsHandler.h File Reference	20
4.5.1 Detailed Description	20
4.6 header/ESC_core/testing_utils.h File Reference	21
4.6.1 Detailed Description	22
4.6.2 Macro Definition Documentation	22
4.6.2.1 FLOAT_TOLERANCE	22
4.6.3 Function Documentation	22
4.6.3.1 expectedErrorNotDetected()	22
4.6.3.2 printGold()	22
4.6.3.3 printGreen()	23
4.6.3.4 printRed()	23
4.6.3.5 testFloatEquals()	23
4.6.3.6 testGreaterThan()	25
4.6.3.7 testGreaterThanOrEqualTo()	26
4.6.3.8 testLessThan()	26
4.6.3.9 testLessThanOrEqualTo()	27
4.6.3.10 testTruth()	28
4.7 source/ESC_core/AssetsManager.cpp File Reference	28
4.7.1 Detailed Description	28
4.8 source/ESC_core/InputsHandler.cpp File Reference	29
4.8.1 Detailed Description	29
4.9 source/ESC_core/testing_utils.cpp File Reference	29
4.9.1 Detailed Description	30
4.9.2 Function Documentation	30
4.9.2.1 expectedErrorNotDetected()	30
4.9.2.2 printGold()	30
4.9.2.3 printGreen()	31
4.9.2.4 printRed()	31
4.9.2.5 testFloatEquals()	31
4.9.2.6 testGreaterThan()	32
4.9.2.7 testGreaterThanOrEqualTo()	32
4.9.2.8 testLessThan()	33
4.9.2.9 testLessThanOrEqualTo()	34

---

4.9.2.10 testTruth()	34
4.10 test/ESC_core/test_AssetsManager.cpp File Reference	35
4.10.1 Detailed Description	35
4.10.2 Function Documentation	35
4.10.2.1 main()	36
4.11 test/ESC_core/test_InputsHandler.cpp File Reference	37
4.11.1 Detailed Description	37
4.11.2 Function Documentation	37
4.11.2.1 main()	37
<b>Bibliography</b>	<b>39</b>
<b>Index</b>	<b>41</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">AssetsManager</a>	
A class which manages visual and sound assets . . . . .	5
<a href="#">InputsHandler</a>	
A class which handles inputs from peripherals (i.e., keyboard and mouse) . . . . .	8





## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

header/ESC_core/ <a href="#">AssetsManager.h</a>	
Header file for the <a href="#">AssetsManager</a> class . . . . .	17
header/ESC_core/ <a href="#">constants.h</a>	
Header file for various constants . . . . .	18
header/ESC_core/ <a href="#">doxygen_cite.h</a>	
Header file which simply cites the doxygen tool . . . . .	18
header/ESC_core/ <a href="#">includes.h</a>	
Header file for various includes . . . . .	19
header/ESC_core/ <a href="#">InputsHandler.h</a>	
Header file for the <a href="#">InputsHandler</a> class . . . . .	20
header/ESC_core/ <a href="#">testing_utils.h</a>	
Header file for various testing utilities . . . . .	21
source/ESC_core/ <a href="#">AssetsManager.cpp</a>	
Implementation file for the <a href="#">AssetsManager</a> class . . . . .	28
source/ESC_core/ <a href="#">InputsHandler.cpp</a>	
Implementation file for the <a href="#">InputsHandler</a> class . . . . .	29
source/ESC_core/ <a href="#">testing_utils.cpp</a>	
Implementation file for various testing utilities . . . . .	29
test/ESC_core/ <a href="#">test_AssetsManager.cpp</a>	
Suite of tests for the <a href="#">AssetsManager</a> class . . . . .	35
test/ESC_core/ <a href="#">test_InputsHandler.cpp</a>	
Suite of tests for the <a href="#">InputsHandler</a> class . . . . .	37



## Chapter 3

# Class Documentation

### 3.1 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)
- void [loadTexture](#) (std::string, std::string)
- void [loadSoundBuffer](#) (std::string, std::string)
- void [loadSound](#) (std::string, std::string)
- void [loadTrack](#) (std::string, std::string)
- [~AssetsManager](#) (void)  
*Destructor for the [AssetsManager](#) class.*

#### Private Attributes

- std::map< std::string, sf::Font \* > [font\\_map](#)
- std::map< std::string, sf::Texture \* > [texture\\_map](#)
- std::map< std::string, sf::SoundBuffer \* > [soundbuffer\\_map](#)
- std::map< std::string, sf::Sound \* > [sound\\_map](#)
- std::map< std::string, sf::Music \* >::iterator [current\\_track](#)
- std::map< std::string, sf::Music \* > [track\\_map](#)

#### 3.1.1 Detailed Description

A class which manages visual and sound assets.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 AssetsManager()

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

Constructor for the [AssetsManager](#) class.

```
46 {
47     //...
48
49     std::cout << "AssetsManager constructed at " << this << std::endl;
50
51     return;
52 } /* AssetsManager() */
```

#### 3.1.2.2 ~AssetsManager()

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

Destructor for the [AssetsManager](#) class.

```
75 {
76     std::cout << "AssetsManager at " << this << " destroyed" << std::endl;
77
78     return;
79 } /* ~AssetsManager() */
```

### 3.1.3 Member Function Documentation

#### 3.1.3.1 loadFont()

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

#### 3.1.3.2 loadSound()

```
void AssetsManager::loadSound (
    std::string ,
    std::string )
```

### 3.1.3.3 loadSoundBuffer()

```
void AssetsManager::loadSoundBuffer (
    std::string ,
    std::string )
```

### 3.1.3.4 loadTexture()

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

### 3.1.3.5 loadTrack()

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

## 3.1.4 Member Data Documentation

### 3.1.4.1 current\_track

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

### 3.1.4.2 font\_map

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

### 3.1.4.3 sound\_map

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

#### 3.1.4.4 soundbuffer\_map

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

#### 3.1.4.5 texture\_map

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

#### 3.1.4.6 track\_map

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

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

- header/ESC\_core/[AssetsManager.h](#)
- source/ESC\_core/[AssetsManager.cpp](#)

## 3.2 InputsHandler Class Reference

A class which handles inputs from peripherals (i.e., keyboard and mouse).

```
#include <InputsHandler.h>
```

### Public Member Functions

- [InputsHandler](#) (void)  
*Constructor for the [InputsHandler](#) class.*
- void [process](#) (sf::Event \*)
- void [printKeysPressed](#) (void)  
*Method to print out which keys are currently pressed.*
- void [reset](#) (void)  
*Method to reset [InputsHandler](#). To be called once per frame (at end of frame!).*
- [~InputsHandler](#) (void)  
*Destructor for the [InputsHandler](#) class.*

### Public Attributes

- std::vector< bool > [key\\_pressed\\_once\\_vec](#)
- std::vector< bool > [key\\_press\\_vec](#)
- std::map< sf::Keyboard::Key, std::string > [key\\_code\\_map](#)

## Private Member Functions

- void `__constructKeyCodeMap` (void)

*Helper method to construct a map from `sf::Keyboard::Key` to a string representation of the corresponding key.*

### 3.2.1 Detailed Description

A class which handles inputs from peripherals (i.e., keyboard and mouse).

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 InputHandler()

```
InputHandler::InputHandler (  
    void )
```

Constructor for the `InputHandler` class.

```
379 {  
380     this->key_pressed_once_vec.resize(sf::Keyboard::KeyCount, false);  
381     this->key_press_vec.resize(sf::Keyboard::KeyCount, false);  
382  
383     this->__constructKeyCodeMap();  
384  
385     std::cout << "InputHandler constructed at " << this << std::endl;  
386  
387     return;  
388 } /* InputHandler() */
```

#### 3.2.2.2 ~InputHandler()

```
InputHandler::~~InputHandler (  
    void )
```

Destructor for the `InputHandler` class.

```
499 {  
500     std::cout << "InputHandler at " << this << " destroyed" << std::endl;  
501  
502     return;  
503 } /* ~InputHandler() */
```

### 3.2.3 Member Function Documentation

### 3.2.3.1 `__constructKeyCodeMap()`

```
void InputsHandler::__constructKeyCodeMap (
    void ) [private]
```

Helper method to construct a map from `sf::Keyboard::Key` to a string representation of the corresponding key.

```
35 {
36     // 1. unknown keys
37     this->key_code_map.insert(
38         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Unknown, "Unknown")
39     );
40
41
42     // 2. alpha keys
43     this->key_code_map.insert(
44         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::A, "A")
45     );
46     this->key_code_map.insert(
47         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::B, "B")
48     );
49     this->key_code_map.insert(
50         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::C, "C")
51     );
52     this->key_code_map.insert(
53         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::D, "D")
54     );
55     this->key_code_map.insert(
56         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::E, "E")
57     );
58     this->key_code_map.insert(
59         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F, "F")
60     );
61     this->key_code_map.insert(
62         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::G, "G")
63     );
64     this->key_code_map.insert(
65         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::H, "H")
66     );
67     this->key_code_map.insert(
68         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::I, "I")
69     );
70     this->key_code_map.insert(
71         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::J, "J")
72     );
73     this->key_code_map.insert(
74         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::K, "K")
75     );
76     this->key_code_map.insert(
77         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::L, "L")
78     );
79     this->key_code_map.insert(
80         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::M, "M")
81     );
82     this->key_code_map.insert(
83         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::N, "N")
84     );
85     this->key_code_map.insert(
86         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::O, "O")
87     );
88     this->key_code_map.insert(
89         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::P, "P")
90     );
91     this->key_code_map.insert(
92         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Q, "Q")
93     );
94     this->key_code_map.insert(
95         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::R, "R")
96     );
97     this->key_code_map.insert(
98         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::S, "S")
99     );
100    this->key_code_map.insert(
101        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::T, "T")
102    );
103    this->key_code_map.insert(
104        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::U, "U")
105    );
106    this->key_code_map.insert(
107        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::V, "V")
108    );
109    this->key_code_map.insert(
110        std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::W, "W")
111    );
112    this->key_code_map.insert(
```



```
113         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::X, "X")
114     );
115     this->key_code_map.insert (
116         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Y, "Y")
117     );
118     this->key_code_map.insert (
119         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Z, "Z")
120     );
121
122
123     // 3. numeric keys
124     this->key_code_map.insert (
125         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num0, "0")
126     );
127     this->key_code_map.insert (
128         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num1, "1")
129     );
130     this->key_code_map.insert (
131         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num2, "2")
132     );
133     this->key_code_map.insert (
134         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num3, "3")
135     );
136     this->key_code_map.insert (
137         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num4, "4")
138     );
139     this->key_code_map.insert (
140         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num5, "5")
141     );
142     this->key_code_map.insert (
143         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num6, "6")
144     );
145     this->key_code_map.insert (
146         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num7, "7")
147     );
148     this->key_code_map.insert (
149         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num8, "8")
150     );
151     this->key_code_map.insert (
152         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Num9, "9")
153     );
154     this->key_code_map.insert (
155         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad0, "0")
156     );
157     this->key_code_map.insert (
158         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad1, "1")
159     );
160     this->key_code_map.insert (
161         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad2, "2")
162     );
163     this->key_code_map.insert (
164         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad3, "3")
165     );
166     this->key_code_map.insert (
167         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad4, "4")
168     );
169     this->key_code_map.insert (
170         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad5, "5")
171     );
172     this->key_code_map.insert (
173         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad6, "6")
174     );
175     this->key_code_map.insert (
176         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad7, "7")
177     );
178     this->key_code_map.insert (
179         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad8, "8")
180     );
181     this->key_code_map.insert (
182         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Numpad9, "9")
183     );
184
185
186     // 4. direction keys
187     this->key_code_map.insert (
188         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Left, "Left")
189     );
190     this->key_code_map.insert (
191         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Right, "Right")
192     );
193     this->key_code_map.insert (
194         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Up, "Up")
195     );
196     this->key_code_map.insert (
197         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Down, "Down")
198     );
199
```

```

200
201 // 5. function keys
202 this->key_code_map.insert (
203     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F1, "F1")
204 );
205 this->key_code_map.insert (
206     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F2, "F2")
207 );
208 this->key_code_map.insert (
209     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F3, "F3")
210 );
211 this->key_code_map.insert (
212     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F4, "F4")
213 );
214 this->key_code_map.insert (
215     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F5, "F5")
216 );
217 this->key_code_map.insert (
218     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F6, "F6")
219 );
220 this->key_code_map.insert (
221     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F7, "F7")
222 );
223 this->key_code_map.insert (
224     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F8, "F8")
225 );
226 this->key_code_map.insert (
227     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F9, "F9")
228 );
229 this->key_code_map.insert (
230     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F10, "F10")
231 );
232 this->key_code_map.insert (
233     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F11, "F11")
234 );
235 this->key_code_map.insert (
236     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F12, "F12")
237 );
238 this->key_code_map.insert (
239     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F13, "F13")
240 );
241 this->key_code_map.insert (
242     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F14, "F14")
243 );
244 this->key_code_map.insert (
245     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::F15, "F15")
246 );
247
248
249 // 6. other keys
250 this->key_code_map.insert (
251     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Escape, "Escape")
252 );
253 this->key_code_map.insert (
254     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LControl, "LCtrl")
255 );
256 this->key_code_map.insert (
257     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LShift, "LShift")
258 );
259 this->key_code_map.insert (
260     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LAlt, "LAlt")
261 );
262 this->key_code_map.insert (
263     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LSystem, "LSystem")
264 );
265 this->key_code_map.insert (
266     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RControl, "RCtrl")
267 );
268 this->key_code_map.insert (
269     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RShift, "RShift")
270 );
271 this->key_code_map.insert (
272     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RAlt, "RAlt")
273 );
274 this->key_code_map.insert (
275     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RSystem, "RSystem")
276 );
277 this->key_code_map.insert (
278     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Menu, "Menu")
279 );
280 this->key_code_map.insert (
281     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::LBracket, "LBracket")
282 );
283 this->key_code_map.insert (
284     std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::RBracket, "RBracket")
285 );
286 this->key_code_map.insert (

```

```

287         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Semicolon, "Semicolon")
288     );
289     this->key_code_map.insert (
290         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Comma, "Comma")
291     );
292     this->key_code_map.insert (
293         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Period, "Period")
294     );
295     this->key_code_map.insert (
296         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Quote, "Quote")
297     );
298     this->key_code_map.insert (
299         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Slash, "Slash")
300     );
301     this->key_code_map.insert (
302         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Backslash, "Backslash")
303     );
304     this->key_code_map.insert (
305         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Tilde, "Tilde")
306     );
307     this->key_code_map.insert (
308         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Equal, "Equal")
309     );
310     this->key_code_map.insert (
311         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Hyphen, "Hyphen")
312     );
313     this->key_code_map.insert (
314         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Space, "Space")
315     );
316     this->key_code_map.insert (
317         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Enter, "Enter")
318     );
319     this->key_code_map.insert (
320         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Backspace, "Backspace")
321     );
322     this->key_code_map.insert (
323         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Tab, "Tab")
324     );
325     this->key_code_map.insert (
326         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::PageUp, "PageUp")
327     );
328     this->key_code_map.insert (
329         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::PageDown, "PageDown")
330     );
331     this->key_code_map.insert (
332         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::End, "End")
333     );
334     this->key_code_map.insert (
335         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Home, "Home")
336     );
337     this->key_code_map.insert (
338         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Insert, "Insert")
339     );
340     this->key_code_map.insert (
341         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Delete, "Delete")
342     );
343     this->key_code_map.insert (
344         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Add, "Add")
345     );
346     this->key_code_map.insert (
347         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Subtract, "Subtract")
348     );
349     this->key_code_map.insert (
350         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Multiply, "Multiply")
351     );
352     this->key_code_map.insert (
353         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Divide, "Divide")
354     );
355     this->key_code_map.insert (
356         std::pair<sf::Keyboard::Key, std::string>(sf::Keyboard::Pause, "Pause")
357     );
358
359     return;
360 } /* __constructKeyCodeMap() */

```

### 3.2.3.2 printKeysPressed()

```

void InputsHandler::printKeysPressed (
    void )

```

Method to print out which keys are currently pressed.

```

448 {
449     std::string print_str = "";
450
451     for (size_t i = 0; i < this->key_press_vec.size(); i++) {
452         if (this->key_press_vec[i]) {
453             print_str += this->key_code_map[sf::Keyboard::Key(i)];
454             print_str += ", ";
455         }
456     }
457
458     if (not print_str.empty()) {
459         std::cout << "Keys pressed: " << print_str << std::endl;
460     }
461
462     return;
463 } /* printKeysPressed() */

```

### 3.2.3.3 process()

```

void InputsHandler::process (
    sf::Event * event_ptr )
{
405 {
406     // 1. update state of key press vectors
407     switch (event_ptr->type) {
408         case (sf::Event::KeyPressed): {
409             if (not this->key_press_vec[event_ptr->key.code]) {
410                 this->key_pressed_once_vec[event_ptr->key.code] = true;
411             }
412
413             this->key_press_vec[event_ptr->key.code] = true;
414
415             break;
416         }
417
418         case (sf::Event::KeyReleased): {
419             this->key_pressed_once_vec[event_ptr->key.code] = false;
420             this->key_press_vec[event_ptr->key.code] = false;
421
422             break;
423         }
424
425         default: {
426             // do nothing!
427
428             break;
429         }
430     }
431
432     return;
433 } /* process() */

```

### 3.2.3.4 reset()

```

void InputsHandler::reset (
    void )

```

Method to reset [InputsHandler](#). To be called once per frame (at end of frame!).

```

478 {
479     for (size_t i = 0; i < this->key_press_vec.size(); i++) {
480         this->key_pressed_once_vec[i] = false;
481     }
482
483     return;
484 } /* reset() */

```

## 3.2.4 Member Data Documentation

### 3.2.4.1 key\_code\_map

```
std::map<sf::Keyboard::Key, std::string> InputsHandler::key_code_map
```

### 3.2.4.2 key\_press\_vec

```
std::vector<bool> InputsHandler::key_press_vec
```

### 3.2.4.3 key\_pressed\_once\_vec

```
std::vector<bool> InputsHandler::key_pressed_once_vec
```

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

- [header/ESC\\_core/InputsHandler.h](#)
- [source/ESC\\_core/InputsHandler.cpp](#)



## File Documentation

Header file for the `AssetsManager` class.

```
graph BT; A[source/ESC_core/AssetsManager.cpp] --> B[header/ESC_core/AssetsManager.h]; C[test/ESC_core/test_AssetsManager.cpp] --> B;
```

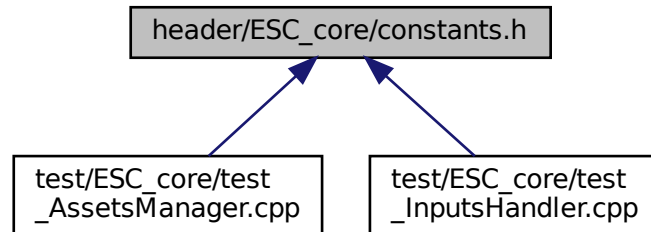
- class `AssetsManager`  
*A class which manages visual and sound assets.*

Header file for the `AssetsManager` class.

## 4.2 header/ESC\_core/constants.h File Reference

Header file for various constants.

This graph shows which files directly or indirectly include this file:



### Variables

- `const int FRAMES_PER_SECOND = 60`
- `const double SECONDS_PER_FRAME = 1.0 / 60`

#### 4.2.1 Detailed Description

Header file for various constants.

#### 4.2.2 Variable Documentation

##### 4.2.2.1 FRAMES\_PER\_SECOND

```
const int FRAMES_PER_SECOND = 60
```

##### 4.2.2.2 SECONDS\_PER\_FRAME

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

## 4.3 header/ESC\_core/doxygen\_cite.h File Reference

Header file which simply cites the doxygen tool.



### 4.3.1 Detailed Description

Header file which simply cites the doxygen tool.

Ref: [van Heesch. \[2023\]](#)

## 4.4 header/ESC\_core/includes.h File Reference

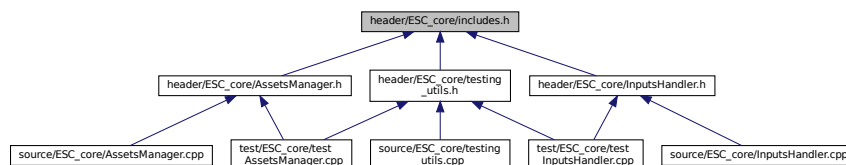
Header file for various includes.

```
#include <cmath>
#include <cstdlib>
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <limits>
#include <list>
#include <map>
#include <stdexcept>
#include <sstream>
#include <string>
#include <vector>
#include <SFML/Audio.hpp>
#include <SFML/Config.hpp>
#include <SFML/GpuPreference.hpp>
#include <SFML/Graphics.hpp>
#include <SFML/Main.hpp>
#include <SFML/Network.hpp>
#include <SFML/OpenGL.hpp>
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
```

Include dependency graph for includes.h:



This graph shows which files directly or indirectly include this file:



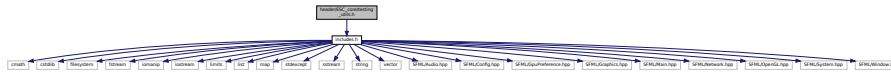


## 4.6 header/ESC\_core/testing\_utils.h File Reference

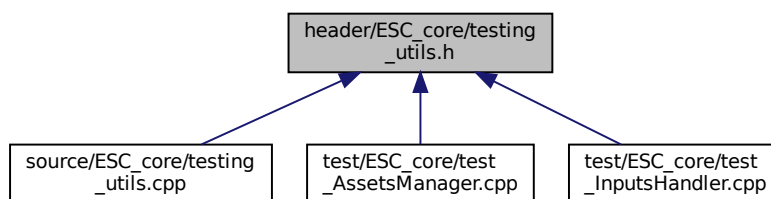
Header file for various testing utilities.

```
#include "includes.h"
```

Include dependency graph for testing\_utils.h:



This graph shows which files directly or indirectly include this file:



### Macros

- `#define` `FLOAT_TOLERANCE` 1e-6  
A tolerance for application to floating point equality tests.

### 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  $x \geq y$ .
- void `testLessThan` (double, double, std::string, int)  
Tests if  $x < y$ .
- void `testLessThanOrEqualTo` (double, double, std::string, int)  
Tests if  $x \leq y$ .
- 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.

### 4.6.1 Detailed Description

Header file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

### 4.6.2 Macro Definition Documentation

#### 4.6.2.1 FLOAT\_TOLERANCE

```
#define FLOAT_TOLERANCE 1e-6
```

A tolerance for application to floating point equality tests.

### 4.6.3 Function Documentation

#### 4.6.3.1 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

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

#### 4.6.3.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

#### Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {  
83     std::cout << "\x1B[33m" << input_str << "\033[0m";  
84     return;  
85 } /* printGold() */
```

#### 4.6.3.3 printGreen()

```
void printGreen (  
    std::string input_str )
```

A function that sends green text to std::cout.

#### Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
62 {  
63     std::cout << "\x1B[32m" << input_str << "\033[0m";  
64     return;  
65 } /* printGreen() */
```

#### 4.6.3.4 printRed()

```
void printRed (  
    std::string input_str )
```

A function that sends red text to std::cout.

#### Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
102 {  
103     std::cout << "\x1B[31m" << input_str << "\033[0m";  
104     return;  
105 } /* printRed() */
```

#### 4.6.3.5 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

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

```

136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";
150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif
156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

## 4.6.3.6 testGreaterThan()

```

void testGreaterThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x > y$ .

## Parameters

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

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif

```

```

207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

#### 4.6.3.7 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x \geq y$ .

##### Parameters

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

```

240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);
260     return;
261 } /* testGreaterThanOrEqualTo() */

```

#### 4.6.3.8 testLessThan()

```

void testLessThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x < y$ .

##### Parameters

<i>x</i>	The first of two numbers to test.
----------	-----------------------------------



## Parameters

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

```

291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);
311     return;
312 } /* testLessThan() */

```

## 4.6.3.9 testLessThanOrEqualTo()

```

void testLessThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x \leq y$ .

## Parameters

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

```

342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;

```





## 4.9.1 Detailed Description

Implementation file for various testing utilities.

This is a library of utility functions used throughout the various test suites.

## 4.9.2 Function Documentation

### 4.9.2.1 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

<i>file</i>	The file in which the test is applied (you should be able to just pass in "__FILE__").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in "__LINE__").

```
430 {
431     std::string error_str = "\n ERROR   failed to throw expected error prior to line ";
432     error_str += std::to_string(line);
433     error_str += " of ";
434     error_str += file;
435
436     #ifdef _WIN32
437         std::cout << error_str << std::endl;
438     #endif
439
440     throw std::runtime_error(error_str);
441     return;
442 } /* expectedErrorNotDetected() */
```

### 4.9.2.2 printGold()

```
void printGold (
    std::string input_str )
```

A function that sends gold text to std::cout.

#### Parameters

<i>input_str</i>	The text of the string to be sent to std::cout.
------------------	---

```
82 {
83     std::cout << "\x1B[33m" << input_str << "\033[0m";
84     return;
85 } /* printGold() */
```

### 4.9.2.3 printGreen()

```
void printGreen (
    std::string input_str )
```

A function that sends green text to `std::cout`.

#### Parameters

<i>input_str</i>	The text of the string to be sent to <code>std::cout</code> .
------------------	---

```
62 {
63     std::cout << "\x1B[32m" << input_str << "\033[0m";
64     return;
65 } /* printGreen() */
```

### 4.9.2.4 printRed()

```
void printRed (
    std::string input_str )
```

A function that sends red text to `std::cout`.

#### Parameters

<i>input_str</i>	The text of the string to be sent to <code>std::cout</code> .
------------------	---

```
102 {
103     std::cout << "\x1B[31m" << input_str << "\033[0m";
104     return;
105 } /* printRed() */
```

### 4.9.2.5 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

<i>x</i>	The first of two numbers to test.
<i>y</i>	The second of two numbers to test.
<i>file</i>	The file in which the test is applied (you should be able to just pass in " <code>__FILE__</code> ").
<i>line</i>	The line of the file in which the test is applied (you should be able to just pass in " <code>__LINE__</code> ").

```
136 {
137     if (fabs(x - y) <= FLOAT_TOLERANCE) {
138         return;
```

```

139     }
140
141     std::string error_str = "ERROR: testFloatEquals():\t in ";
142     error_str += file;
143     error_str += "\tline ";
144     error_str += std::to_string(line);
145     error_str += ":\t\n";
146     error_str += std::to_string(x);
147     error_str += " and ";
148     error_str += std::to_string(y);
149     error_str += " are not equal to within +/- ";
150     error_str += std::to_string(FLOAT_TOLERANCE);
151     error_str += "\n";
152
153     #ifdef _WIN32
154         std::cout << error_str << std::endl;
155     #endif
156
157     throw std::runtime_error(error_str);
158     return;
159 } /* testFloatEquals() */

```

#### 4.9.2.6 testGreaterThan()

```

void testGreaterThan (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x > y$ .

##### Parameters

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

```

189 {
190     if (x > y) {
191         return;
192     }
193
194     std::string error_str = "ERROR: testGreaterThan():\t in ";
195     error_str += file;
196     error_str += "\tline ";
197     error_str += std::to_string(line);
198     error_str += ":\t\n";
199     error_str += std::to_string(x);
200     error_str += " is not greater than ";
201     error_str += std::to_string(y);
202     error_str += "\n";
203
204     #ifdef _WIN32
205         std::cout << error_str << std::endl;
206     #endif
207
208     throw std::runtime_error(error_str);
209     return;
210 } /* testGreaterThan() */

```

#### 4.9.2.7 testGreaterThanOrEqualTo()

```

void testGreaterThanOrEqualTo (
    double x,

```

```
double y,
std::string file,
int line )
```

Tests if  $x \geq y$ .

#### Parameters

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

```
240 {
241     if (x >= y) {
242         return;
243     }
244
245     std::string error_str = "ERROR: testGreaterThanOrEqualTo():\t in ";
246     error_str += file;
247     error_str += "\tline ";
248     error_str += std::to_string(line);
249     error_str += ":\t\n";
250     error_str += std::to_string(x);
251     error_str += " is not greater than or equal to ";
252     error_str += std::to_string(y);
253     error_str += "\n";
254
255     #ifdef _WIN32
256         std::cout << error_str << std::endl;
257     #endif
258
259     throw std::runtime_error(error_str);
260     return;
261 } /* testGreaterThanOrEqualTo() */
```

#### 4.9.2.8 testLessThan()

```
void testLessThan (
    double x,
    double y,
    std::string file,
    int line )
```

Tests if  $x < y$ .

#### Parameters

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

```
291 {
292     if (x < y) {
293         return;
294     }
295
296     std::string error_str = "ERROR: testLessThan():\t in ";
297     error_str += file;
298     error_str += "\tline ";
299     error_str += std::to_string(line);
300     error_str += ":\t\n";
```

```

301     error_str += std::to_string(x);
302     error_str += " is not less than ";
303     error_str += std::to_string(y);
304     error_str += "\n";
305
306     #ifdef _WIN32
307         std::cout << error_str << std::endl;
308     #endif
309
310     throw std::runtime_error(error_str);
311     return;
312 } /* testLessThan() */

```

#### 4.9.2.9 testLessThanOrEqualTo()

```

void testLessThanOrEqualTo (
    double x,
    double y,
    std::string file,
    int line )

```

Tests if  $x \leq y$ .

##### Parameters

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

```

342 {
343     if (x <= y) {
344         return;
345     }
346
347     std::string error_str = "ERROR: testLessThanOrEqualTo():\t in ";
348     error_str += file;
349     error_str += "\tline ";
350     error_str += std::to_string(line);
351     error_str += ":\t\n";
352     error_str += std::to_string(x);
353     error_str += " is not less than or equal to ";
354     error_str += std::to_string(y);
355     error_str += "\n";
356
357     #ifdef _WIN32
358         std::cout << error_str << std::endl;
359     #endif
360
361     throw std::runtime_error(error_str);
362     return;
363 } /* testLessThanOrEqualTo() */

```

#### 4.9.2.10 testTruth()

```

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

```

Tests if the given statement is true.



## Parameters

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

```

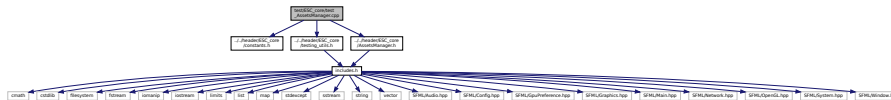
390 {
391     if (statement) {
392         return;
393     }
394
395     std::string error_str = "ERROR: testTruth():\t in ";
396     error_str += file;
397     error_str += "\tline ";
398     error_str += std::to_string(line);
399     error_str += ":\t\t\n";
400     error_str += "Given statement is not true";
401
402     #ifdef _WIN32
403         std::cout << error_str << std::endl;
404     #endif
405
406     throw std::runtime_error(error_str);
407     return;
408 } /* testTruth() */

```

#### 4.10 test/ESC\_core/test\_AssetsManager.cpp File Reference

Suite of tests for the [AssetsManager](#) class.

```
#include "../..header/ESC_core/constants.h"
#include "../..header/ESC_core/testing_utils.h"
#include "../..header/ESC_core/AssetsManager.h"
Include dependency graph for test_AssetsManager.cpp:
```



## Functions

- `int main (int argc, char **argv)`

#### 4.10.1 Detailed Description

Suite of tests for the [AssetsManager](#) class.

A suite of tests for the `AssetsManager` class.

### 4.10.2 Function Documentation

## 4.10.2.1 main()

```

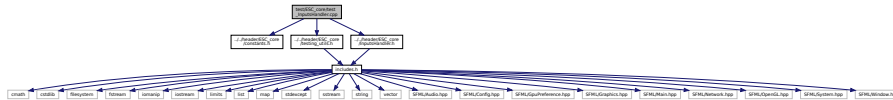
int main (
    int argc,
    char ** argv )
36 {
37     #ifdef _WIN32
38         activateVirtualTerminal();
39     #endif /* _WIN32 */
40
41     printGold("\tTesting AssetsManager");
42     std::cout << std::endl;
43
44     srand(time(NULL));
45     int n_dots = 8;
46
47
48     try {
49         //...
50
51         sf::Clock clock;
52         sf::Event event;
53         sf::RenderWindow window(sf::VideoMode(800, 600), "Testing AssetsManager");
54
55         unsigned long long int frame = 0;
56         double time_since_run_s = 0;
57
58         while (window.isOpen()) {
59             time_since_run_s = clock.getElapsedTime().asSeconds();
60
61             if (
62                 time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
63             ) {
64                 while (window.pollEvent(event))
65                 {
66                     //...
67
68                     if (event.type == sf::Event::Closed) {
69                         window.close();
70                     }
71                 }
72
73                 window.clear();
74                 window.display();
75
76                 //...
77
78                 std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
79                 frame++;
80             }
81         }
82     }
83
84
85     catch (...) {
86         //...
87
88         printGold(" ");
89         for (int i = 0; i < n_dots; i++) {
90             printGold(".");
91         }
92         printGold(" ");
93         printRed("FAIL");
94         std::cout << std::endl;
95         throw;
96     }
97
98
99     //...
100
101     printGold(" ");
102     for (int i = 0; i < n_dots; i++) {
103         printGold(".");
104     }
105     printGold(" ");
106     printGreen("PASS");
107     std::cout << std::endl;
108
109     return 0;
110 } /* main() */

```

#### 4.11 test/ESC\_core/test\_InputsHandler.cpp File Reference

Suite of tests for the `InputsHandler` class.

```
#include "../..//header/ESC_core/constants.h"
#include "../..//header/ESC_core/testing_utils.h"
#include "../..//header/ESC_core/InputsHandler.h"
Include dependency graph for test_InputsHandler.cpp:
```



## Functions

- `int main (int argc, char **argv)`

#### 4.11.1 Detailed Description

Suite of tests for the `InputsHandler` class.

A suite of tests for the `InputsHandler` class.

### 4.11.2 Function Documentation

#### 4.11.2.1 main()

```

1  int main (
2
3      int argc,
4
5      char ** argv )
6
7
8
9
10
11
12
13
14
15
16 {
17
18     #ifdef _WIN32
19         activateVirtualTerminal();
20     #endif /* _WIN32 */
21
22
23     printGold("\tTesting InputsHandler");
24     std::cout << std::endl;
25
26
27     srand(time(NULL));
28     int n_dots = 8;
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49     try {
50
51         InputsHandler inputs_handler;
52
53
54         testFloatEquals(
55             int(sf::Keyboard::KeyCount),
56             101,
57             __FILE__,
58             __LINE__
59         );
60
61         testFloatEquals(
62             inputs_handler.key_press_vec.size(),
63             int(sf::Keyboard::KeyCount),
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

61         __FILE__,
62         __LINE__
63     );
64
65     testFloatEquals(
66         inputs_handler.key_pressed_once_vec.size(),
67         int(sf::Keyboard::KeyCount),
68         __FILE__,
69         __LINE__
70     );
71
72     sf::Clock clock;
73     sf::Event event;
74     sf::RenderWindow window(sf::VideoMode(800, 600), "Testing InputsHandler");
75
76     unsigned long long int frame = 0;
77     double time_since_run_s = 0;
78
79     while (window.isOpen()) {
80         time_since_run_s = clock.getElapsedTime().asSeconds();
81
82         if (
83             time_since_run_s >= (frame + 1) * SECONDS_PER_FRAME
84         ) {
85             while (window.pollEvent(event))
86             {
87                 inputs_handler.process(&event);
88
89                 if (event.type == sf::Event::Closed) {
90                     window.close();
91                 }
92             }
93
94             window.clear();
95             window.display();
96
97             //inputs_handler.printKeysPressed();
98             if (inputs_handler.key_pressed_once_vec[sf::Keyboard::Enter]) {
99                 std::cout << "Enter" << std::endl;
100             }
101
102             inputs_handler.reset();
103
104             std::cout << frame << " : " << time_since_run_s << "\r" << std::flush;
105             frame++;
106         }
107     }
108 }
109
110 catch (...) {
111     //...
112
113     printGold(" ");
114     for (int i = 0; i < n_dots; i++) {
115         printGold(".");
116     }
117     printGold(" ");
118     printRed("FAIL");
119     std::cout << std::endl;
120     throw;
121 }
122
123 //...
124
125 printGold(" ");
126 for (int i = 0; i < n_dots; i++) {
127     printGold(".");
128 }
129 printGold(" ");
130 printGreen("PASS");
131 std::cout << std::endl;
132
133 return 0;
134 }
135 /* main() */
136 }

```

# Bibliography

L. Gomila. SFML: Simple and Fast Multimedia Library, 2023. URL <https://www.sfml-dev.org/>. 20

D. van Heesch. Doxygen: Generate documentation from source code, 2023. URL <https://www.doxygen.nl>. 19



# Index

- [\\_\\_constructKeyCodeMap](#)
    - [InputsHandler, 9](#)
  - [~AssetsManager](#)
    - [AssetsManager, 6](#)
  - [~InputsHandler](#)
    - [InputsHandler, 9](#)
- [AssetsManager, 5](#)
  - [~AssetsManager, 6](#)
  - [AssetsManager, 6](#)
  - [current\\_track, 7](#)
  - [font\\_map, 7](#)
  - [loadFont, 6](#)
  - [loadSound, 6](#)
  - [loadSoundBuffer, 6](#)
  - [loadTexture, 7](#)
  - [loadTrack, 7](#)
  - [sound\\_map, 7](#)
  - [soundbuffer\\_map, 7](#)
  - [texture\\_map, 8](#)
  - [track\\_map, 8](#)
- [constants.h](#)
  - [FRAMES\\_PER\\_SECOND, 18](#)
  - [SECONDS\\_PER\\_FRAME, 18](#)
- [current\\_track](#)
  - [AssetsManager, 7](#)
- [expectedErrorNotDetected](#)
  - [testing\\_utils.cpp, 30](#)
  - [testing\\_utils.h, 22](#)
- [FLOAT\\_TOLERANCE](#)
  - [testing\\_utils.h, 22](#)
- [font\\_map](#)
  - [AssetsManager, 7](#)
- [FRAMES\\_PER\\_SECOND](#)
  - [constants.h, 18](#)
- [header/ESC\\_core/AssetsManager.h, 17](#)
- [header/ESC\\_core/constants.h, 18](#)
- [header/ESC\\_core/doxygen\\_cite.h, 18](#)
- [header/ESC\\_core/includes.h, 19](#)
- [header/ESC\\_core/InputsHandler.h, 20](#)
- [header/ESC\\_core/testing\\_utils.h, 21](#)
- [InputsHandler, 8](#)
  - [\\_\\_constructKeyCodeMap, 9](#)
  - [~InputsHandler, 9](#)
  - [InputsHandler, 9](#)
  - [key\\_code\\_map, 15](#)
  - [key\\_press\\_vec, 15](#)
  - [key\\_pressed\\_once\\_vec, 15](#)
  - [printKeysPressed, 13](#)
  - [process, 14](#)
  - [reset, 14](#)
- [key\\_code\\_map](#)
  - [InputsHandler, 15](#)
- [key\\_press\\_vec](#)
  - [InputsHandler, 15](#)
- [key\\_pressed\\_once\\_vec](#)
  - [InputsHandler, 15](#)
- [loadFont](#)
  - [AssetsManager, 6](#)
- [loadSound](#)
  - [AssetsManager, 6](#)
- [loadSoundBuffer](#)
  - [AssetsManager, 6](#)
- [loadTexture](#)
  - [AssetsManager, 7](#)
- [loadTrack](#)
  - [AssetsManager, 7](#)
- [main](#)
  - [test\\_AssetsManager.cpp, 35](#)
  - [test\\_InputsHandler.cpp, 37](#)
- [printGold](#)
  - [testing\\_utils.cpp, 30](#)
  - [testing\\_utils.h, 22](#)
- [printGreen](#)
  - [testing\\_utils.cpp, 30](#)
  - [testing\\_utils.h, 23](#)
- [printKeysPressed](#)
  - [InputsHandler, 13](#)
- [printRed](#)
  - [testing\\_utils.cpp, 31](#)
  - [testing\\_utils.h, 23](#)
- [process](#)
  - [InputsHandler, 14](#)
- [reset](#)
  - [InputsHandler, 14](#)
- [SECONDS\\_PER\\_FRAME](#)
  - [constants.h, 18](#)
- [sound\\_map](#)
  - [AssetsManager, 7](#)
- [soundbuffer\\_map](#)
  - [AssetsManager, 7](#)

- source/ESC\_core/AssetsManager.cpp, 28
- source/ESC\_core/InputsHandler.cpp, 29
- source/ESC\_core/testing\_utils.cpp, 29
  
- test/ESC\_core/test\_AssetsManager.cpp, 35
- test/ESC\_core/test\_InputsHandler.cpp, 37
- test\_AssetsManager.cpp
  - main, 35
- test\_InputsHandler.cpp
  - main, 37
- testFloatEquals
  - testing\_utils.cpp, 31
  - testing\_utils.h, 23
- testGreaterThan
  - testing\_utils.cpp, 32
  - testing\_utils.h, 25
- testGreaterThanOrEqualTo
  - testing\_utils.cpp, 32
  - testing\_utils.h, 26
- testing\_utils.cpp
  - expectedErrorNotDetected, 30
  - printGold, 30
  - printGreen, 30
  - printRed, 31
  - testFloatEquals, 31
  - testGreaterThan, 32
  - testGreaterThanOrEqualTo, 32
  - testLessThan, 33
  - testLessThanOrEqualTo, 34
  - testTruth, 34
- testing\_utils.h
  - expectedErrorNotDetected, 22
  - FLOAT\_TOLERANCE, 22
  - printGold, 22
  - printGreen, 23
  - printRed, 23
  - testFloatEquals, 23
  - testGreaterThan, 25
  - testGreaterThanOrEqualTo, 26
  - testLessThan, 26
  - testLessThanOrEqualTo, 27
  - testTruth, 28
- testLessThan
  - testing\_utils.cpp, 33
  - testing\_utils.h, 26
- testLessThanOrEqualTo
  - testing\_utils.cpp, 34
  - testing\_utils.h, 27
- testTruth
  - testing\_utils.cpp, 34
  - testing\_utils.h, 28
- texture\_map
  - AssetsManager, 8
- track\_map
  - AssetsManager, 8