**§ 9.2 – EventHandler**

This class is responsible for all event handling. Every class that derives from **BaseEngineInterface** contains an event handler that can be used.

The main purpose is for handling events.

It contains a set of methods that can be called to directly respond to events, though internally these are used to called custom callbacks that are registered. These are as follows:

void HandleMouseMovement(const sf::Vector2i &v)

* Invoked to respond to mouse movement

void HandleMouseOver(const sf::Vector2i &v)

* Invoked to respond to mouse over events

void HandleMouseExit(const sf::Vector2i &v)

* Invoked to respond to mouse exit events

void HandleMousePress(const sf::Vector2i &v, const sf::Mouse::Button &b)

* Invoked to respond to mouse press events

void HandleMouseRelease(const sf::Vector2i &v, const sf::Mouse::Button &b)

* Invoked to respond to mouse release events

void HandleMouseScroll(const sf::Vector2i &v)

* Invoked to respond to mouse scroll events

void HandleKeyPress(const sf::Keyboard::Key &k)

* Invoked to respond to key press events
* This is only triggered once per press by default, but a config can be set so that the OS-delay is allowed and events can be fired continuously

void HandleKeyRelease(const sf::Keyboard::Key &k)

* Invoked to respond to key release events
* This is only ever triggered once per press/release cycle per key (but it will trigger for every key, even if multiple keys are pressed)

void HandleTextEntered(const sf::Keyboard::Key &k)

* Invoked to respond to “text entered” events
* This is useful for text input, and it will automatically respond using the OS-delay and the key code can be directly appended to a string

The event handler has a set of function pointers contained internally that are called inside its corresponding function listed previously to allow for custom behavior to be bound. The function pointers are as follows:

std::function<void(const sf::Vector2i &)> ftnCallback\_MouseMovement;

std::function<void(const sf::Vector2i &)> ftnCallback\_MouseOver;

std::function<void(const sf::Vector2i &)> ftnCallback\_MouseExit;

std::function<void(const sf::Vector2i &, const sf::Mouse::Button &)> ftnCallback\_MousePress;

std::function<void(const sf::Vector2i &, const sf::Mouse::Button &)> ftnCallback\_MouseRelease;

std::function<void(const sf::Vector2i &)> ftnCallback\_MouseScroll;

std::function<void(const sf::Keyboard::Key &)> ftnCallback\_KeyPress;

std::function<void(const sf::Keyboard::Key &)> ftnCallback\_KeyRelease;

std::function<void(const sf::Keyboard::Key &)> ftnCallback\_TextEntered;

std::function<void(void)> ftnCallback\_WindowClosed;

std::function<void(void)> ftnCallback\_WindowResized;

These can be bound by using the method void BindCallback

There are 4 version of the method:

void BindCallback(const Events &type, std::function<void(const sf::Vector2i &)> ftn)

* Used to bind the functions that take only a sf::Vector2i as a parameter
* Those are:
  + HandleMouseMovement
  + HandleMouseOver
  + HandleMouseExit
  + HandleMouseScroll

void BindCallback(const Events &type, std::function<void(const sf::Vector2i &, const sf::Mouse::Button &)> ftn)

* Used to bind the functions that take a sf::Vector2i and a sf::Mouse::Button as a parameter.
* Those are:
  + HandleMousePress
  + HandleMouseRelease

void BindCallback(const Events &type, std::function<void(const sf::Keyboard::Key &)> ftn)

* Used to bind the functions that take a sf::Keyboard::Key as a parameter.
* Those are:
  + HandleKeyPress
  + HandleKeyRelease
  + HandleTextEntered

void BindCallback(const Events &type, std::function<void(void)> ftn)

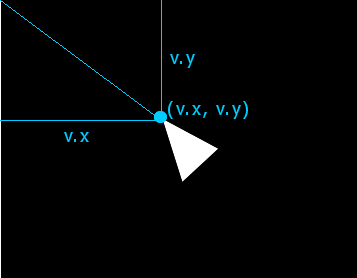
* Used to bind the functions that take no parameters
* Those are:
  + ftnCallback\_WindowClosed (This has no corresponding member method)
  + ftnCallback\_WindowResized (This has no corresponding member method)

**How each is/can be used:**

Each method is used for a specific purpose. Here are how they are used:

**void HandleMouseMovement(const sf::Vector2i &v)**

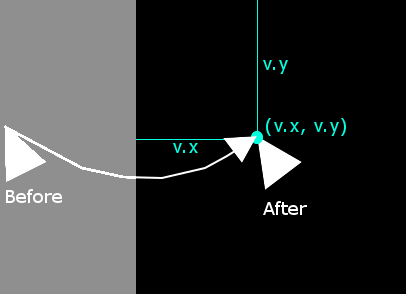
Whenever the mouse is moved inside of an object, this method is called. Passed as a parameter is the coordinate on which the mouse is currently at relative to the upper left-hand corner of the object’s collision box.



This will be called every time the mouse moves *inside* the object. If the mouse moves on top of the object from being outside the object, HandleMouseOver is called. If the mouse moves out of the object’s region, HandleMouseExit is called.

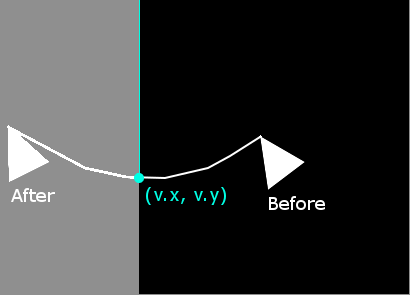
**void HandleMouseOver(const sf::Vector2i &v)**

This is called whenever the mouse moves over an object after before being outside the object’s bounds. Passed as a parameter is the position the mouse is currently at.



**void HandleMouseExit(const sf::Vector2i &v)**

This is basically the opposite of HandleMouseOver. This is triggered whenever the mouse leaves the object. Passed as a parameter is the position along the edge where the exiting mouse would have hit were it still inside the object.



**void HandleMousePress(const sf::Vector2i &v, const sf::Mouse::Button &b)**

This will be called whenever the mouse is pressed on an object. Passed as parameters are:

* The position of the mouse, relative to the upper left-hand corner (as in HandleMouseMovement)
* The code of the mouse button (see <http://www.sfml-dev.org/documentation/2.4.1/classsf_1_1Keyboard.php#acb4cacd7cc5802dec45724cf3314a142>)

**void HandleMouseRelease(const sf::Vector2i &v, const sf::Mouse::Button &b)**

This will be called whenever the mouse is released on an object.

**Examples of Usage:**

Here are some examples of using the event handler class.

Scenario 1 – Responding to a mouse movement

Consider a class that wishes to display the mouse’s position in text and update it whenever the mouse moves.

MyClass::MyClass()

{

//Bind the 'MouseMoved' event to change the text to something like

//MousePos: (1, 3)

Handler.BindCallback(Events::MouseMoved,

[this](const sf::vector2i &v)

{

this->SetText(“MousePos: (" + std::to\_string(v.x) + ", " +

std::to\_string(v.y) + ")");

});

Handler.BindCallback(Events::MouseOver,

[this](const sf::vector2i &v)

{

this->SetText(“MousePos: (" + std::to\_string(v.x) + ", " +

std::to\_string(v.y) + ")");

});

Handler.BindCallback(Events::MouseExit,

[this](const sf::vector2i &v)

{

this->SetText(“Mouse not over");

});

SetText("Mouse not over");

}