To use the event system the following needs to be done:

Assumptions: Below the variable e will be used to represent a pointer to the EventSystemHandler.

This can be easily gotten by doing the following:

EventSystemHandler\* e = EventSystemHandler::getInstance();

Also inside of your part of the code that is observing the event, make sure that if any part of that event is needed to persist then do a deep copy / manually save all the information you need as the event is deleted afterwards.

**Case 1** - The event I want to observe or emit already exists:

**Case 1.1** - I want to emit / tell others that this event occurred.

Event\* ev = new MyEvent();

e->emitEvent(ev);

**Case 1.2** - I want to observe an event:

Create a MethodObserver<EVENT, CURRENTCLASS> object inside of the class;

e.g. MethodObserver<ButtonEvent, TestClass> mo(this, TestClass::testMethod);

Create a Method in your class that takes in a pointer to the EVENT and returns void

e.g. void testMethod(ButtonEvent \*e);

Register with the eventSystem handler in the following way:

TestClass\* testVar = new TestClass();

EventSystemHandler::getInstance->addObserver(&((testVar)->methodObserverObject),   
 EventTypes::BUTTON);

**Case 2** - The event I want to Observe or Emit does not exitst:

**1** - Create a new class named MyClassNameEvent.

e.g. ButtonEvent.cpp, and ButtonEvent.h

**2 -** This new class must inherit from Event.

e.g. class ButtonEvent : public Event{}

**3 -** Add the event to the enum inside of the EventTypes.h file.

e.g. enum EventType {  
 BUTTON,

EVENTS\_NUM

};

**4 -** This new class must implement: EventTypes::EventType getType();

e.g. EventTypes::EventType ButtonEvent::getType(){

return EventTypes::BUTTON;

}

**5 -** The event is now created, see Case 1 for further instructions.