# GDD

## Features

The event listener class has a preset list of available key inputs (which can be extended if needed), and dispatches the event to all listeners waiting to hear the chosen event. When the dispatch fires, the set function for the command is executed, producing a response.

### **Conditions of satisfaction**

* #include “FLInputManager.h”
* Have your desired class inherit from the EventListener class. This is required to detect the key input.
* Get an instance of InputManager as follows: InputManager\* inputManager = InputManager::getInstance(); this returns the same input manager across all classes.
* This input manager is used to map keys, by using the AddKey() function.
* In your update loop, you need: inputManager->ProcessInput(); this is required to run the dispatch.

### Manual Test

* Add a key using the input manager AddKey()
* Process Input in the update loop.
* Run the application.
* Press any key set up in the input manager.
* If the key has a listener or function, it will run when pressed.

## **Event Listener**

* When you add a key using AddKey() it creates an event listener, that takes in a specific input.
* When that input is received, process it, and dispatch the desired event.
* For every listener, if the listener event was called, return the event to each listener.
* By using an override of the onEvent() you can call functions upon dispatches, without binding them to keys.

## **Key / Controller Button Pressed/ Released/ Held**

### **Conditions of satisfaction**

* The inputmanager processinput() function updates an Event.
* A key value is sent in.
* The input type, and the input key is detected, and dispatched by the event listener for execution.
* By Using SetControllerButtonDelay() you can set the repeat rate for the controller buttons.

## **Mouse Buttons / Wheel**

* The inputmanager processinput() function updates an Event.
* A button / wheel value is sent in.
* The input type (Press and Release, no hold), and the input key is detected, and dispatched by the event listener for execution.

## **Controller Sticks**

* The input manager takes in an update of the sticks.
* The sticks return a vector / normal / angle.

## **Controller Triggers**

* The input manager takes in an update of the triggers.
* The triggers operate with a value between -32000 and 32000.
* The input manager may return the soft or full hard press, and a release.
* You may also get the triggers exact value.
* By Using SetControllerTriggerDelay() you can set the repeat rate for the controller buttons.

## **Custom Commands**

* A custom Command object is made for each command for the application.
* This command is assigned to any key available.
* When the key is executed, the command it has been assigned is executed.

## **Input Logger**

* A log file is created with the time and date in the project directory.
* Each key, type, duration and time is logged.
* All commands logged in CSV file.

## **Reset Commands**

* Clear custom commands from keys, by specifying the event.

## **Vector2f**

* Takes in an x and a y variable.
* Can return the magnitude, dot product and normal.
* Custom \* and + operator to multiply and add vectors.