|  |  |
| --- | --- |
| **GAT150 – Introduction to Game Programming** |  |

***Input System***

***OVERVIEW***

In this assignment you will create an input system to enable interactivity with our program.

***GRADING***

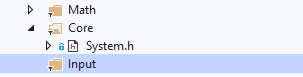
This assignment is worth 50 points. To receive full credit, your engine must include the input system and accept input from the keyboard to control a sprite. Submit the InputSystem.h and InputSystem.cpp for the submission.

***INSTRUCTIONS***

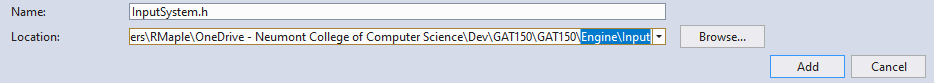
**There are parts of code where there is <…>. These are parts you have to fill in.**

# Create the Input System class

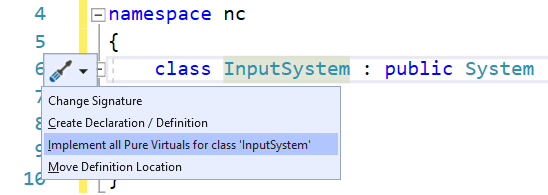
* Create an *Input* filter in *Engine*



* Create an *InputSystem.h* in *Engine/Input*



* Add an *InputSystem* class in *InputSystem.h*
  + Derive the *InputSystem* from *System*
  + You can auto create the abstract functions of the *System* in *InputSystem*
    - Click on InputSystem and press (ctrl + .), then select *Implement all Pure Virtuals…*



* + - Make sure the functions are public
* This will create an *InputSystem.cpp*, drag that into the *Input* filter



# Information on how to get Keyboard states

*This is provided so you’ll understand the code later*

* To get the keyboard input, the function SDL\_GetKeyboardState is used
  + <https://wiki.libsdl.org/SDL_GetKeyboardState>
  + This function returns a pointer to an array of keyboard states stored in SDL
    - If the state is 1 then the key is pressed, 0 not pressed
  + The function takes a parameter of a pointer to an int which retrieves the number of keyboard states, it can also be passed nullptr if the number of keys is not requested

const Uint8\* SDL\_GetKeyboardState(int\* numkeys)

# Add the Keyboard Input members and method declarations

* We will store the current keyboard state and the previous keyboard state in an array that will be dynamically allocated (new)
* We will also have a member that is the numbers of keys in our array
  + Add protected members to the class

Uint8\* m\_keystate;

Uint8\* m\_prevKeystate;

int m\_numKeys;

* Create an enum *eButtonState* of the button states (IDLE, PRESSED, HELD, RELEASED)
  + Declare these at the top of the class and make it public

public:

enum class eButtonState

{

IDLE,

PRESSED,

HELD,

RELEASED

};

* Declare the following three methods
  + The int id will be the SDL\_Scancode
    - * <https://wiki.libsdl.org/SDL_Scancode>

eButtonState GetButtonState(int id);

bool GetButtonDown(int id);

bool GetPreviousButtonDown(int id);

* Create the methods in the .cpp (highlight the methods and press (ctrl + .))

# Add the method definitions

* Read this page on the C function *memcpy* <https://www.tutorialspoint.com/c_standard_library/c_function_memcpy.htm>
* **Startup()**

// get current keystate and retrieve numKeys

const Uint8\* keystate = SDL\_GetKeyboardState(<pointer to m\_numKeys>);

// allocate memory for current and previous keystate

m\_keystate = new <type of elements>[<size of array>];

m\_prevKeystate = new <type of elements> [<size of array>];

// copy current keystate (source) to m\_keystate (destination)

memcpy(<destination>, <source>, <size to copy>);

// copy current keystate to m\_prevKeystate

memcpy(<destination>, <source>, <size to copy>);

return true;

* **Shutdown()**
  + Delete the allocated *m\_keystate* and *m\_prevKeystate*
    - They were allocated as arrays, use the delete[]
* **Update()**

// copy m\_keystate (source) to m\_prevKeystate (destination)

memcpy(<destination>, <source>, <size to copy>);

// get current keystate

const Uint8\* keystate = SDL\_GetKeyboardState(nullptr);

// copy current keystate to m\_keystate

memcpy(<destination>, <source>, <size to copy>);

* **GetButtonDown(int id)**
  + Return the keystate at m\_keystate[id]
* **GetPreviousButtonDown(int id)**
  + Return the keystate at m\_prevKeystate[id]
* **GetButtonState(int id)**

**eButtonState state = eButtonState::IDLE;**

**bool buttonDown = GetButtonDown(id);**

**bool prevButtonDown = GetPreviousButtonDown(id);**

**if (buttonDown)**

**{**

**state = (prevButtonDown) ? ??? : ???;**

**}**

**else**

**{**

**state = (prevButtonDown) ? ??? : ???;**

**}**

**return state;**

* + Use the table to fill in the enum values (**???**) for the button states
    - *buttonDown* is the Current State
    - *preButtonDown* is the Previous State

|  |  |  |
| --- | --- | --- |
| **State** | **Previous State** | **Current State** |
| **IDLE** | **0** | **0** |
| **PRESSED** | **0** | **1** |
| **HELD** | **1** | **1** |
| **RELEASED** | **1** | **0** |

# Add the input code for main()

* Create the input manager in main, don’t forget to #include

nc::InputSystem inputSystem;

* Call Startup() on the inputSystem where the other systems are startup
* Call Update() on the inputSystem in the while loop
* Call Shutdown() on the inputSystem at the bottom where the other systems are shutdown
* Declare a Vector2 position {400, 300} after the float angle declaration
* Utilize the position with one of the textures

texture1->Draw(**position**, { 1, 1 }, angle);

* In the update loop, update the position with the input GetButtonState
  + Use the enum in InputSystem for the HELD state

if (inputSystem.GetButtonState(SDL\_SCANCODE\_LEFT) == <**HELD>**)

{

position.x = position.x - 1.0f;

}

if (inputSystem.GetButtonState(SDL\_SCANCODE\_RIGHT) == <**HELD>**)

{

position.x = position.x + 1.0f;

}

*Test if the input system is working correctly. If you can move the sprite to the left and right of the screen then the input system is functioning correctly. Submit the InputSystem.h and InputSystem.cpp for submission.*

***RESOURCES***

<https://wiki.libsdl.org/SDL_GetKeyboardState>

<https://wiki.libsdl.org/SDL_Scancode>