INPUT HANDLING

Reading Input

- For our player to be more interactive, we will need some way for the user to control the player on screen
- For this we can either read key presses or mouse movements to tell the player cube what to do and where to go
- Reading input can be achieved anywhere in the game code, and is handled through the *Input Manager*. For this we need to include the following header file :

```
#include "InputManager.h"
```

- ***Note : The Handmade Game Engine supports both keyboard input and mouse input***
- ***Note : We will perform all of our input handling within our game objects, specifically from within their Update() routines***

Keyboard Input

- To determine if a key has been pressed, use the *IsKeyPressed()* function. This will return a *bool* value based on if a key is *up* or *down*
- Ideally, this function can be used in a *if-else* statement, like so :

```
if (TheInput::Instance()->IsKeyPressed())
{
    //a key was pressed
}

else
{
    //a key was released
}
```

Keyboard Input

- The above function is sometimes too generic, as it never tells us which key was pressed or released.
- For more specific keyboard input, we need to determine which exact key was pressed
- Internally the *Input Manager* stores an array which maintains which keys are pressed and which are not
- We need to get this array from the *Input Manager* and for this we first need to declare the following pointer :

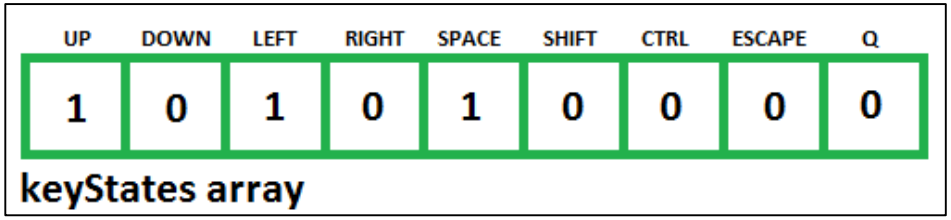
```
const Uint8* keyStates;
```

Keyboard Input

- Now we can call the *GetKeyStates()* function to acquire the array of key states and store it in our pointer :

```
keyStates = TheInput::Instance()->GetKeyStates();
```

- Internally, a small portion of the *keyStates* array may look like this :



- ***Note : Each array element represents a key and for each element with the value 1, a key is pressed and 0 means that the key is released.***

Keyboard Input

- Using the above array in an *if-statement* and indexing it using constant values that represent each key, we can determine if that key was pressed or not.
- If the index value queried returns a **1** (or *true*), the key is pressed, if it returns **0** (or *false*) it is not pressed

```
if (keyStates[SDL_SCANCODE_ESCAPE])
{
    //the ESCAPE key was pressed
}
```

- ***Note : For a complete list of supported key codes, click on the link below :***

  https://wiki.libsdl.org/SDL_Scancode

Mouse Input

- We can also read mouse motion and clicks and for this we also make use of the *Input Manager*
- To see if any particular mouse button is pressed or released we can use either of the *GetButtonState()* functions, like so :

```
if (TheInput::Instance()->GetLeftButtonState() == InputManager::DOWN)
{
    //left mouse button is clicked
}

if (TheInput::Instance()->GetRightButtonState() == InputManager::UP)
{
    //right mouse button is released
}

if (TheInput::Instance()->GetMiddleButtonState() == InputManager::DOWN)
{
    //middle mouse button is clicked
}
```

Mouse Input

- To see how much the mouse has moved, we use the *GetMouseMotion()* routine and store the returned value in a *vec2* object

```
glm::vec2 mouseMotion = TheInput::Instance()->GetMouseMotion();
```

- The x and y values returned correspond to how much the mouse has moved on its x (*left/right*) and y (*up/down*) axis
- The x mouse motion value will be ***negative*** for *left* and ***positive*** for *right* movement
- The y mouse motion value will be ***negative*** for *up* and ***positive*** for *down* movement
- ***Note : The more rigorously you move the mouse, the higher the returned values will be***

Note!!!

- When using mouse motion, because the main camera makes use of it, you will end up rotating the main viewing camera **AND** your player cube
- To change this, simply make sure **ONLY** the following code is in the *MainCamera.cpp* constructor :

```
//set initial position and rotation
m_camera.Position().x = 0;
m_camera.Position().y = 5;
m_camera.Position().z = 5;
m_camera.RotateY(0);
m_camera.RotateX(30);

//disable mouse cursor so that it does not interfere when rotating the camera
TheInput::Instance()->SetMouseCursorState(InputManager::OFF);
```

Note!!!

- Also make sure only the following is in the *MainCamera.cpp Update()* routine :

```
//store keyboard key states in a temp variable for processing below
const Uint8* keyState = TheInput::Instance()->GetKeyStates();

//if ESCAPE key was pressed, return flag to end game
if (keyState[SDL_SCANCODE_ESCAPE])
{
    return false;
}

return true;
```

- Now we have simplified the main camera so that it has a static position and does not move when we move the mouse or press the keys