Lab 2 – Input

1. Building on the work from last week, create and add an Input class (Input.h/.cpp) to the project. This Input class should handle keyboard and mouse input similar to the one discussed in the lecture. Begin by having the class store key presses.
2. Write an application to test your Input class. Have the program output message boxes when:
    a. The 'W' key is pressed.
    b. When the three keys 'J', 'K' and 'L' are pressed together (but not individually).
    c. Have the application exit/close when the Escape key is pressed.
3. Add to your Input class so it stores the mouse's current position. Once done, write an application that renders (in the top left corner) the mouse's current position. This should update as the mouse moves around the window. (see below for information on text rendering).
4. Add to your Input class so it stores the state of mouse button clicks (both left and right button). Then write an application that measures and outputs the distance of a mouse drag (while holding the left mouse button down). Output as a message box on completion of drag or rendered text in the window. You can either calculate the distance the mouse moved by separately calculating the difference in X and Y positions, or use Pythagoras to calculate the distance.
5. Write an application that on mouse right-click spawns and renders a circle at the mouse's location. On subsequent clicks the circle should be rendered at the mouse's new/current position.

Text rendering

Download the font file from Blackboard and extract the "font" folder into the project directory. Before rendering text, you must first load the font required. *Font* is a "sf::Font" variable.

```
if (!font.loadFromFile("font/arial.ttf"))
{
	// something went wrong
	MessageBox(NULL, L"Font failed to load", L"Error", MB_OK);
}
```

The above code will attempt to load the font file, if it fails it will display an error message. Once loaded, the text variable can be configured. Below, *text* is a sf::Text variable, and is configured with the loaded in font, with a set size and string ouput. Position and other parameters can be set if required. The text variable can then be rendered in a similar method to simple geometry.

```
text.setString("Hello world");
text.setFont(font);
text.setCharacterSize(30);
text.setFillColor(sf::Color::Red);
```