**Spike:** Task 24

**Title:** Sound Board

**Author:** Sam Huffer, 101633177

# Goals / deliverables:

* Display a background image that can be toggled on or off with the 0 key.
* Load another image with three identifiable sub-regions (tiles) within it to serve as a sprite sheet.
* Define three rectangles that specify the sub-region for each tile’s image
* Display each tile’s image to a unique random location using a toggle on or off in response to the 1, 2 and 3 number keys.

# Technologies, Tools, and Resources used:

* Visual Studio 2019
* Microsoft Word
* Paint
* SDL2
* Online Resources
  1. Prerequisite game management content: <https://www.youtube.com/watch?v=ATa_joa6Gzg&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=1>
  2. Prerequisite timer content: <https://www.youtube.com/watch?v=z9U-Jif4RVU&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=2>
  3. Prerequisite game entity content: <https://www.youtube.com/watch?v=DI6q73p3rEI&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=3>
  4. Rendering images: <https://www.youtube.com/watch?v=bKiejuOaJtU&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=4>
  5. Handling images with asset manager: <https://www.youtube.com/watch?v=UPZol-0fn08&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=5>
  6. Using sprite sheets: <https://www.youtube.com/watch?v=k70aBlef-20&list=PLhJr2LOK-xwxQlevIZ97ZABLw72Eu9he7&index=6>

# Tasks undertaken:

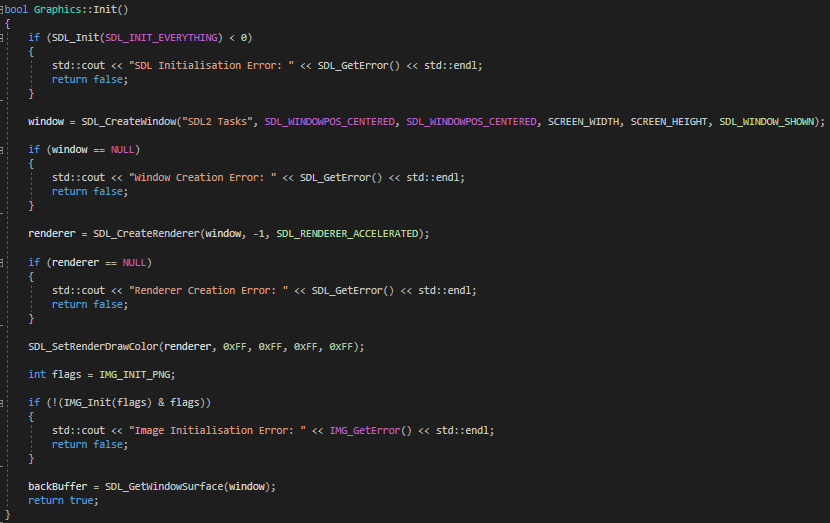
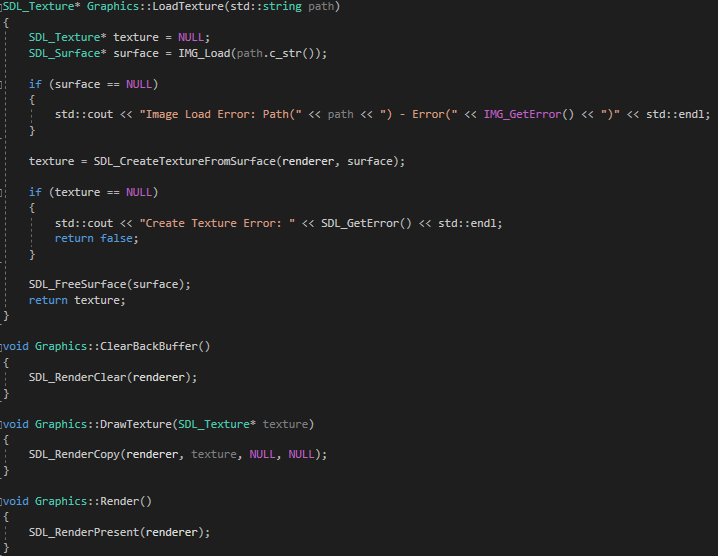
* I copied the task 23 spike report into the task folder, stripping out the spike report’s original content and replacing it with goals and resources pertaining to the task at hand.
* I found an image on my computer to use as the window background for this task, and several other images that could comprise a sprite sheet. For the former, I opened it up in paint, then saved it in “Assets/Images” as a .bmp file. For the latter, I lined them all up in Word, then selected them all and copied them to Paint, before saving them as a .bmp in the same folder.
* I had a look at the YouTube playlist I got the SDL\_Mixer tutorial from to see what I could find that was relevant for this task, and found several tutorials for handling images. However, when I looked at it, it had some established classes where I wasn’t sure if I would need some of their content later on or where I should put the image-related content in my current structure. Therefore, I had a look at some earlier tutorials in the series and reorganised and added basic game management (res. 1), time management (res. 2) and game entity (res. 3) functionality in line with them to get my VS project in line with where the example project is for the image rendering tutorials while also retaining existing functionality in the events polling loop for playing audio files from keystrokes.

Figure : Graphics.Init()

Figure : Graphic’s methods LoadTexture(), ClearBackBuffer(), DrawTexture() and Render().

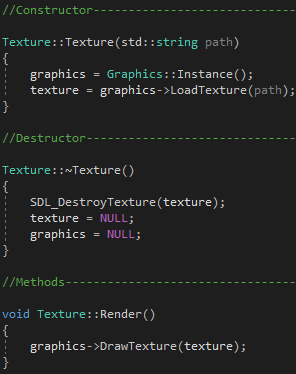
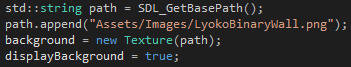
* As part of the prerequisite tutorials, I created a Graphics class to handle loading and rendering graphics, but so far it hadn’t really done anything beyond configure the window when it was initialised. Here, I updated it to also initialise a renderer and the image library as part of its own initialisation (fig. 1).
* Following res. 4 and this task’s objectives, I added to Graphics a LoadTexture() method to allow the loading of images into the program from a file, with appropriate error checking, as well as ClearBackBuffer(), DrawTexture() and Render() to properly render loaded images to the screen.

Figure 4: the lines in GameManager.GameManager() for loading the background.

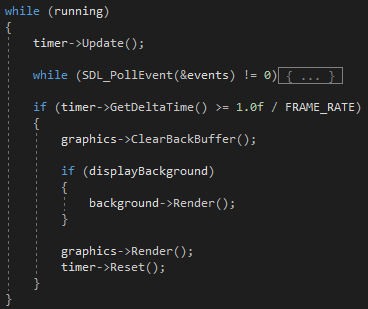
* I added a Texture class that encapsulates SDL\_Texture, and can make calls to Graphics to load and draw itself (fig. 3).
* I added to GameManager a Texture\* field background and updated its constructor to load up an image to be the background (fig. 4). I then added in the rendering section of GameManager.Run()’s game loop a call to background.Render() encapsulated in an if statement checking if the background is to be displayed (fig. 5). I then added to the keyboard input switch a case for the 0 key to toggle the relevant boolean value, changing the keys mapped to other cases in that switch to make room for it.

Figure 3: Implementation of the Texture class.

# What we found out:

Figure 5: the game loop of GameManager.Run(), focusing on the rendering section.

* More about how child game object local and world positions and rotations are calculated.
* More about how time scale is calculated.
* How to load image files into a game using SDL.
* How to render images to the screen using SDL.
* Using the “inline” keyword in a .h file when declaring a method or operator seems to allow you to implement it there rather than in a .cpp file.