**Spike:** 18

**Title:** Game Resource Management (Loading)

**Author:** Ben Holmes, 103024841

**Goals / deliverables:**

Create a simple application, using SDL2, that demonstrates the following features.

1. Keys 1, 2, and 3 will each play a unique sample sound as soon as each key is pressed even if that sound is  
   already playing.
2. Play or pause (not stop) background music in response to key-down press “0” (zero) being used as a toggle.

Create a graphical 2D application capable of displaying images. Your application must:

1. Display a single image as the background image for your application, which can be toggled “on” or “off”  
   using the “0” (zero) key
2. Load one other image that contains three identifiable sub regions (tiles) within it
3. Define three rectangles that specify the sub-region (“part”) for each tiles image
   1. Display each tiles 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:**

* SDL2
* SDL2\_mixer
* Visual Studio 2022
* Word
* <https://lazyfoo.net/tutorials/SDL/04_key_presses/index.php>
* <https://lazyfoo.net/tutorials/SDL/21_sound_effects_and_music/index.php>
* <https://lazyfoo.net/tutorials/SDL/11_clip_rendering_and_sprite_sheets/index.php>

**Tasks undertaken:**

* The music doesn’t play automatically (I could easily but I thought it being started by 0 then paused and unpaused by 0 was nicer)
* Get SDL\_mixer (and image) setup
* Make SDL use relative pathing for the projects so can be used on other machines easily (including lab 17’s code in that
* Get sounds from internet
* Make sound basic structure
* Make sound segment work
* Make graphical sections structure
* Get the sprite sheet rendering properly
* Get the toggling for the sprite sheet working
* Get the toggling for the background working

**What we found out:**

Setting up SDL mixer and image was very easy after the previous lab, basically doing the same steps over with a different library.

The tricky part was realising that I may have to use relative pathing given that I will not be able to demo the SDL spikes so they will have to be marked straight from my github, which could cause problems for running the application on other devices (the library and include dependencies would be broken)

So I added the libraries to the github in the parent folder under the SDLlibs folder and edited all of the project variables for the SDLlibs.

All that would need to be done to run the program now is have a system PATH variable for the SDL modules, and it will run (tested on another machine and works)

A black screen with white text

Description automatically generated

I also setup two files for the different application and setup the main to be able to uncomment and comment them in.

A screenshot of a computer program

Description automatically generated

First of the deliverables was the sound program.

First I created the basic structure of the file, using 4 functions

SDLsound() which is the main function that’s run with the input selection done there.

Init() which initalises the window

Load\_media() which loads the sounds from the files (and a background for nicer look)

Close() which deletes and closes all of the used SDL variables.

I then went on the internet to grab 4 sounds, stored in the resourceSound folder, all from Freesound with correct licencing.

A screenshot of a computer

Description automatically generated

I then made the Load\_media(function)

A computer screen shot of a program

Description automatically generated

Which simply goes through the background and them each sound and loads them from file, turning success to false if failed and displaying error.

Next was the init function:

A computer screen shot of text

Description automatically generated

Which initialises SDL and SDL mixer, then creates the window and its surface.

Third was the close function

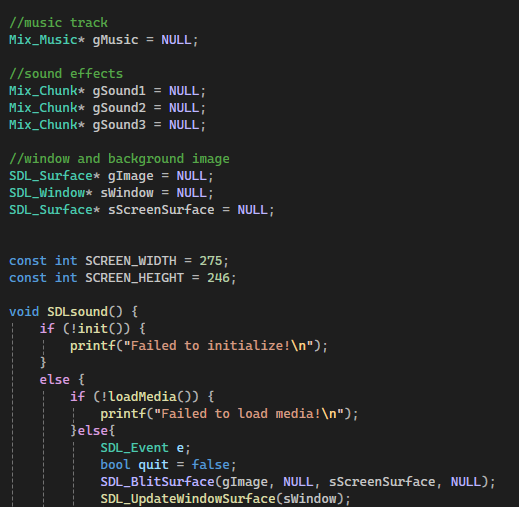
A computer screen shot of a program code

Description automatically generated

Which frees all variables and quits the two SDLs used

And last was the SDLsound (main) function

Staring part:



The Screen\_width and Screen\_height are the hardcoded size of the background image I used.

The loop:

A computer screen shot of a program code

Description automatically generated

This controls the input with 1,2,3 playing the respective sounds and 0 playing and pause/unpausing the guitar music.

The sound system was very simple and logical to interact with and the lazyFoo tutorials made this part easy

A screenshot of a computer

Description automatically generated

(not to scale)

A black screen with white text

Description automatically generated

The second deliverable was the graphical side.

It has an almost identical structure to the sound one (with graph in front of all the identical function names) with an added function:

renderSprite(int x, int y, SDL\_Rect\* clip), which renders one of the sprites from the sprite sheet.

First was the creation of the structure and then I copied across and adjusted the sound code to partially work. I then created the functionality

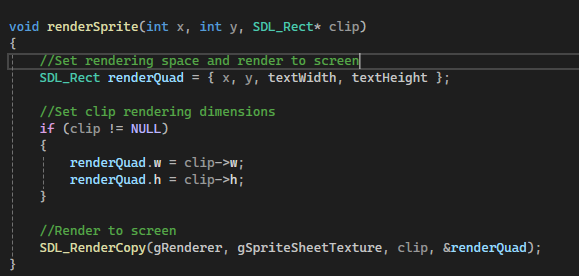
Global variables:

A screen shot of a computer program

Description automatically generated

There are a lot of global variables, each of the sprites needed 3, one to tell if they should be rendered, and two to tell what location they should be rendered at (x and y). the backgroundRend was the check if it should be rendered or not, and the gSpriteClips[3] is the storage of the 3 sprites rectangles.

renderSprite:



This takes in the location, the rectangle of one of the sprites and sets the sprite to render in the correct location one the renderer has been refreshed.

Init:

A screenshot of a computer program

Description automatically generated

Create the window and renderer

Loadmedia:

A computer code on a black background

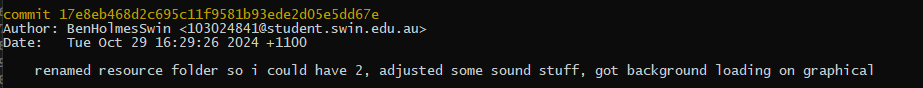
Description automatically generated

A screen shot of a computer program

Description automatically generated

It grabs the images as surfaces and then loads them into textures. It then creates the random x and y (within the screen range) and the dimensions of the sprites on the sprite sheet.

It then free’s (deletes) the surfaces as they are no longer needed



Close()

A computer screen shot of text

Description automatically generated

Destroys and clears all the textures, window and rendererl.

SDLgraphical()

A computer screen with text

Description automatically generated

A computer screen shot of a program code

Description automatically generated

The switch statement for the keys, it flips the Boolean for each relevant render and then outputs whether its coming ON or OFF in the console

A screen shot of a computer program

Description automatically generated

(this is at the end of the while loop)

Clears the render and then adds the renders for each of the sprites and background if then should be, then rerenders.

I had some difficulty with this one as I setup the background like I did in the sound app and lab17 which did not work well with the renderer. I also had the sprites randomising their location on press rather than toggling off and on as I misunderstood the requirements initially.

This made it take a little longer but it was not hard.

A computer screen shot of a program code

Description automatically generated