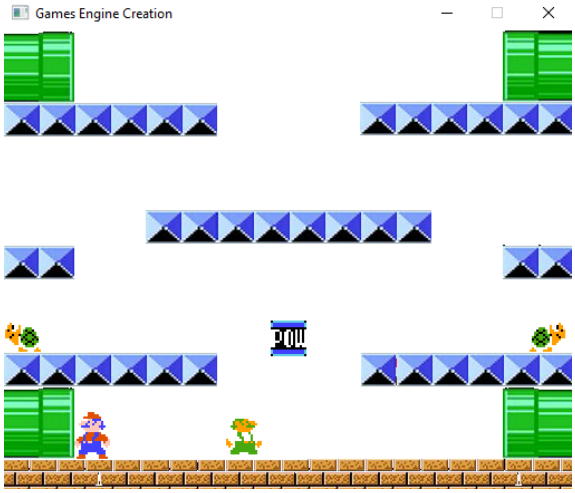
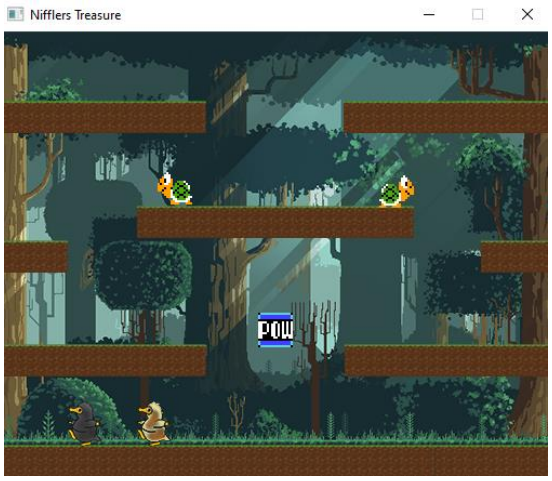
Super Mario Bros

This semester we will be creating the Super Mario Bros, but more specifically we will be creating a lightweight game engine. Although the tutorials are based around Mario, by simply changing the assets we could have a very different game.

These tutorials will only get you so far. You will have to use the knowledge required in semester 1 to help you and you will have to conduct research if you wish to progress above a 40%, these tutorials simply guide/instruct you to create the following:



Provided you complete your accompanying documents and demonstrations, getting to the stage above in your development will get you a pass. As mentioned before making some slight asset changes can give you a different game, like so:



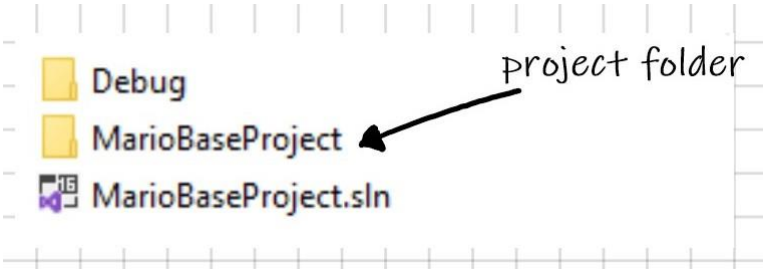
You are quite welcome to create your own assets and own game, it **must** contain the elements taught in the tutorials. Moving beyond the tutorials, in order to gain higher marks and push your

skills, you are on your own. In the past students have created Super Mario 3 and full side scrolling games just to give you an idea of how to gain a high mark.

To create our game, we will be using the Simple DirectMedia Layer (SDL) framework found here: SDL via the official website or there is a zip folder on Blackboard of SDL2 x64. This is the current stable version as of December 2020. SDL is a highly popular framework and has had many titles built off it, your skill and will to learn is the only limit in what you can develop this year…well that and the time frame of the semester.

Tutorial 1 – Setting up SDL

1. Download the SDL files from either Blackboard or the link above. Note: if you are downloading from the link you will also need to find and download the SDL_mixer files along with the SDL_image files. All are contained in the Blackboard folder for you.
2. Open Visual Studio and create a new blank project.
3. Once created, locate the project in the file directory and place the unzipped SDL folder here. *The project folder is the second folder of the same name as your solution where the cpp files are kept. The outer folder that contains the solution is the solution folder i.e. Documents\MarioBaseProject(solution folder)\MarioBaseProject(project folder).*



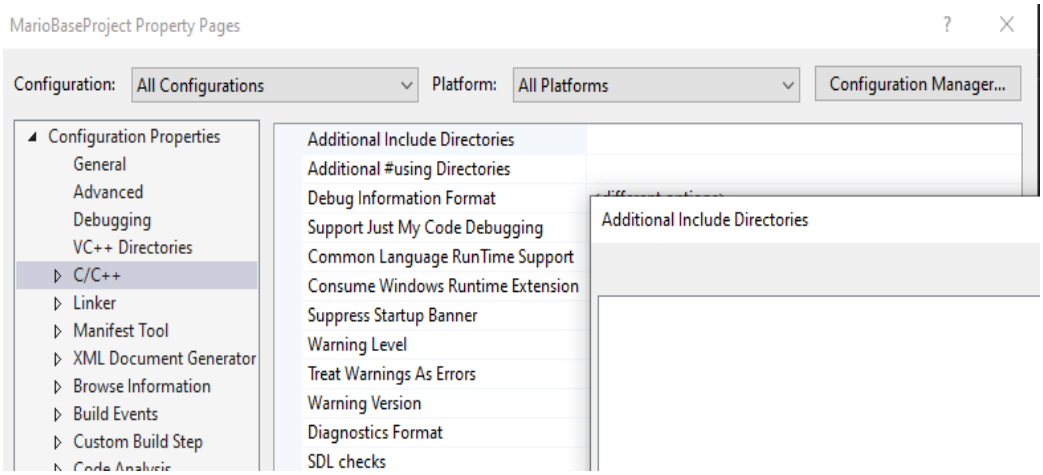
4. In the debug folder within the solution folder (seen above, and not to be mistaken with the debug folder in the project folder) copy all the dll files in the SDL debug folder to this location
5. Back in VS add a new source file and replicate the following code:

```
#include <SDL.h>
#include <SDL_image.h>
#include <SDL_mixer.h>

int main(int argc, char* args[])
{
	return 0;
}
```

Ignore the errors in the includes we will fix them next. As for the main, you may notice it is different from what you are used to in semester 1. Don't worry too much about the meaning but in a nutshell, they are the argument count and argument values passed to the main when it is executed.

6. Now the errors, go to Project -> MarioBaseProject properties and make the following changes:
   a. Select C/C++ -> Additional Include Directories



   and enter the following:

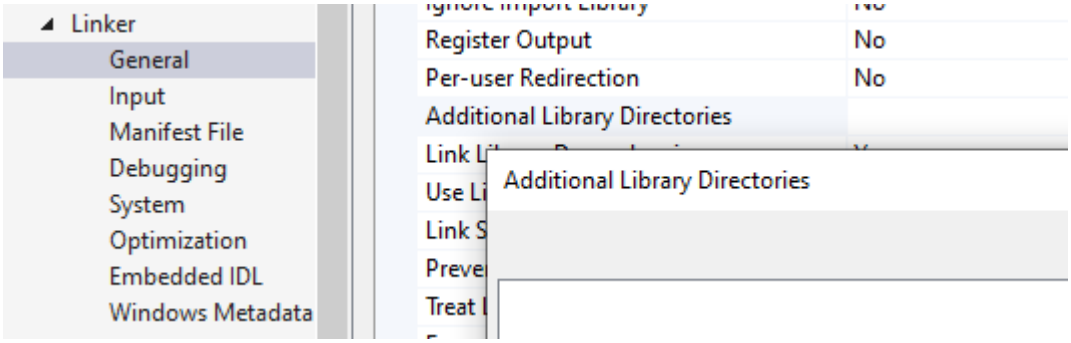
   $(PROJECTDIR)\SDL2\SDL\mixer_include

   $(PROJECTDIR)\SDL2\SDL\image_include

   $(PROJECTDIR)\SDL2\SDL\include

   As you can see the evaluated values is a full directory path. We could have entered this in; however, the project would not work on another machine. Click Apply once done.

   b. Next select Linker->General -> Additional Library Directories



Add the following: $(PROJECTDIR)\SDL2\SDL

c. Finally ensure that Linker -> System -> Subsystem is set to Console (SUBSYSTEM:CONSOLE). Click Apply and OK.

7. Now right click the solution, Add -> Exisiting Item and select all the SDL lib files from the SDL folder.
8. Build and run the program and evverythinng should work.
    a. Any issues go over the steps again to check you have followed the steps correctly.

SDL is now ready to use! I advise that before each tutorial you push to GitHub and create a new branch to work on each time, merging as you go. This will not only be good practice as it is an industry standard, but will also help should anything go wrong and you need to revert to a certain version.