Lab1

Download the Lab1 started solution from GCULEarn.

Create two classes. Right click your project (**Labs**) then click **add** and select **new item** then select **C++ class**. Call one Display and the other MainGame. The header files are as follows...

```
#pragma once
#include <SDL/SDL.h>
#include <GL\glew.h>
#include <iostream>
#include <string>
using namespace std;


class Display
{
public:
	Display();
	~Display();
	void initDisplay();
	void swapBuffer();

private:

	void returnError(std::string errorString);

	SDL_Window* _window; //holds pointer to out window
	int _screenWidth;
	int _screenHeight;
};



#pragma once

#include <SDL\SDL.h>
#include <GL/glew.h>
#include "Display.h"

enum class GameState{PLAY, EXIT};

class MainGame
{
public:
	MainGame();
	~MainGame();

	void run();

private:

	void initSystems();
	void processInput();
	void gameLoop();
	void drawGame();

	Display _gameDisplay;
	GameState _gameState;

};
```

Now open up the main, inside the main we simply want to create a new main game and call its run function...

```
#include <iostream>
#include "MainGame.h"

int main(int argc, char** argv) //arguments used to call SDL main
{
	MainGame mainGame;
	mainGame.run();

	return 0;
}
```

Now open up Display.cpp and paste the following...

```
#include "Display.h"


Display::Display()
{
}

Display::~Display()
{
}

void Display::returnError(std::string errorString)
{
}

void Display::swapBuffer()
{
}

void Display::initDisplay()
{
}
```

TODO

constructer:

- set our window to equal a nullpointer for debugging reasons (= nullptr;)
- set the screen width to = 1024
- set the screen height to = 768

returnError:

- write a simple method that:
  - takes in a string as an argument and prints it to the screen
  - asks the user to "press any key to quit..."
  - takes in a key stroke

- quits SDL (SDL_Quit();) after the key has been pressed

swapBuffer:

- swap the buffer of our window (SDL_GL_SwapWindow())

initDisplay:

- initialise SDL (SDL_Init())
- set up the double buffer (SDL_GL_SetAttribute())
- create our window (SDL_CreateWindow()) using the following arguments "Game Window", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, _screenWidth, _screenHeight, SDL_WINDOW_OPENGL
- Create an SDL_GLContext using SDL_GL_CreateContext, pass the pointer to our window as an argument.
- Initalise GLEW
  - GLenum error = glewInit();
- Check everything initialised, if not use our returnError method to return an error
  - if (_window == nullptr) call returnError and create a suitable message
  - if (glContext == nullptr) call returnError and create a suitable message
  - if (error != GLEW_OK) call returnError and create a suitable message
- Set background colour
  - glClearColor(0.0f, 1.0f, 1.0f, 1.0f);

Now open up MainGame.cpp and paste the following...

```
#include "MainGame.h"
#include <iostream>
#include <string>

MainGame::MainGame()
{
}

MainGame::~MainGame()
{
}

void MainGame::run()
{
}

void MainGame::initSystems()
{
}

void MainGame::gameLoop()
{
}

void MainGame::processInput()
{
}

void MainGame::drawGame()
```

```
{

        // old code for testing only
        glEnableClientState(GL_COLOR_ARRAY);
        glBegin(GL_TRIANGLES);
        glColor3f(1.0f, 0.0f, 0.0f);
        glVertex2f(0, 0);
        glVertex2f(0, 500);
        glVertex2f(500, 500);
        glEnd();

        // SWAP BUFFER HERE
}
```

TODO

constructer:

- set the gamestate to GameState::PLAY
- create new display, remember to make it a pointer Display*

run:

- initialise systems
- run the game loop

initSystems:

- initialise the game display

gameLoop:

- while (_gameState != GameState::EXIT)
    - process inputs
    - draw the game

processInput:

- Create an SDL_event
- Get and process the events while(SDL_PollEvent(&evnt))
- Write a switch statement with a single case switch (evnt.type)
    - case SDL_QUIT then GameState::EXIT;

drawGame:

- set the clear-depth to 1.0 glClearDepth()
- call glclear using the follow arguments GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT
- swap the buffers

Your program should now draw a crude triangle to the screen.