

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  #include <GL/glew.h>
6  #include <GLFW/glfw3.h>
7  #include <glm/glm.hpp>
8
9  #include "Graphics.h"
10
11 Graphics::Graphics() {
12
13 };
14
15 Graphics::~Graphics() {
16
17 };
18
19 int Graphics::Init() {
20     if (!glfwInit()) {                // Checking for GLFW
21         cout << "Could not initialise GLFW...";
22         return 1;
23     }
24
25     glfwSetErrorCallback(ErrorCallbackGLFW);    // Setup a function to catch
26                                                  // and display all GLFW errors.
27
28     glfwHint(GLFW_XXX, XXX);                // Setup glfw with various
29                                             // hints.
30
31     string title = "My OpenGL Application";
32     window = glfwCreateWindow(windowWidth, windowHeight, title.c_str(), NULL,
33                                NULL);
34     if (!window) {                        // Window or OpenGL context
35         cout << "Could not initialise GLFW...";
36         endProgram();
37         return 1;
38     }
39
40     glfwMakeContextCurrent(window);        // making the OpenGL context
41                                             // current
42
43     // Start GLEW (note: always
44     // initialise GLEW after creating your window context.)
45     glewExperimental = GL_TRUE;           // hack: catching them all -
46     forcing newest debug callback (glDebugMessageCallback)
47     GLenum errGLEW = glewInit();
48     if (GLEW_OK != errGLEW) {             // Problems starting GLEW?
49         cout << "Could not initialise GLEW...";
50         endProgram();
51         return 1;
52     }
53 }
```

```
47     }
48
49     SetupRender();
50
51     return 0;
52 }
53
54 void Graphics::hintsGLFW() {
55     glfwWindowHint(GLFW_OPENGL_DEBUG_CONTEXT, GL_TRUE);           // Create
56     context in debug mode - for debug message callback
57     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
58     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
59 }
60 void ErrorCallbackGLFW(int error, const char* description) {
61     cout << "Error GLFW: " << description << "\n";
62 }
63
64
65 void Graphics::endProgram() {
66     glfwMakeContextCurrent(window);           // destroys window handler
67     glfwTerminate();           // destroys all windows and releases resources.
68 }
69
70 void Graphics::SetupRender() {
71     glfwSwapInterval(1);           // Only render when synced (V SYNC)
72
73     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
74     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
75     glfwWindowHint(GLFW_SAMPLES, 0);
76     glfwWindowHint(GLFW_STEREO, GL_FALSE);
77 }
78
79 void Graphics::SetOptimisations() {
80     glEnable(GL_CULL_FACE);
81     glFrontFace(GL_CCW);
82
83     glEnable(GL_DEPTH_TEST);
84     glDepthFunc(GL_LEQUAL);
85 }
86
87 void Graphics::ClearViewport() {
88     glViewport(0, 0, windowWidth, windowHeight);
89     static const GLfloat silver[] = { 0.9f, 0.9f, 0.9f, 1.0f };
90     glClearBufferfv(GL_COLOR, 0, silver);
91     static const GLfloat one = 1.0f;
92     glClearBufferfv(GL_DEPTH, 0, &one);
93 }
94
95
96
```