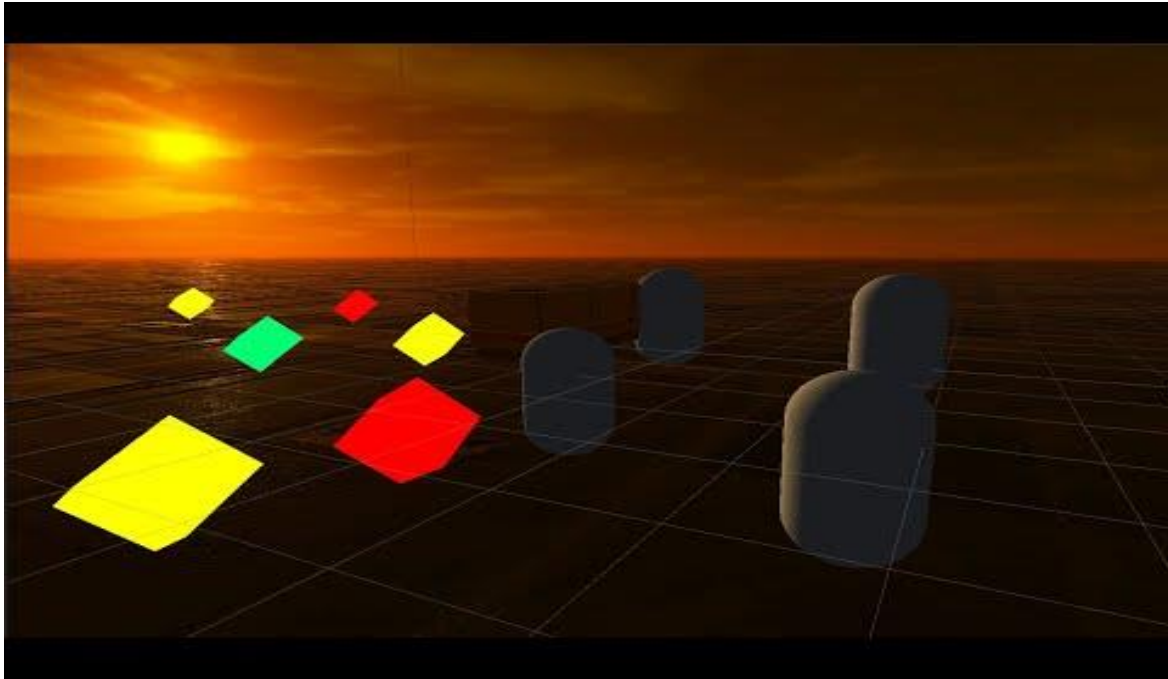


CSC8507 – Technology Implementation Appendices

YouTube Link for our Game Play Demo

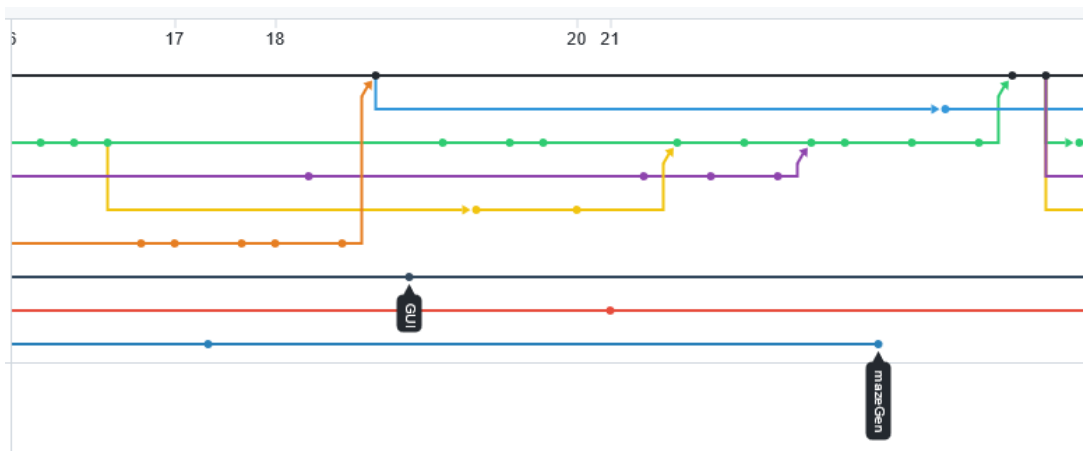
[CSC8507 Splatooth 2042 Prototype](#)



Appendix A – GitHub Commit Logs

Copy of Team 6's GitHub commit logs from 23/02/2022

Commits on Feb 23, 2022		
Merge branch 'FMODImp'	909fad9	<>
David Wong (PGT) committed 9 minutes ago		
Commits on Feb 22, 2022		
Update CMAKE	fc8cc89	<>
DavidWong85 committed 12 hours ago		
Commits on Feb 21, 2022		
Merged Every Unity branches except GUI	32b91c7	<>
hkj293 committed 2 days ago		
Merge branch 'UnityGraphics'	070733b	<>
hkj293 committed 2 days ago		
Merged MazeGen	5033207	<>
hkj293 committed 2 days ago		
Update FMOD c++	1ae9112	<>
DavidWong85 committed 2 days ago		
Added Collectable Gen	1eea302	<>
hkj293 committed 2 days ago		
Merged FMOD	cdf13cc	<>
hkj293 committed 2 days ago		
Merge branch 'UnityFMOD' into UnityGraphics	6796c36	<>
hkj293 committed 2 days ago		
update	c624Fc2	<>
DavidWong85 committed 2 days ago		
Merged SceneManager	0446959	<>
hkj293 committed 2 days ago		
fmod update	1b69904	<>
DavidWong85 committed 2 days ago		
Merge branch 'SceneManager' into UnityGraphics	2dbea4d	<>
hkj293 committed 2 days ago		
add sound	ea933ec	<>
DavidWong85 committed 2 days ago		



Appendix B – Jenkins Build Logs

Copy of Team 6's Jenkins commit logs from 23/02/2022

The screenshot shows the Jenkins dashboard for a project named 'Team6GroupProject'. The left sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes' (selected), 'Workspace', 'Build Now', 'Configure', 'Delete Project', 'Git Polling Log', and 'Rename'. Below these is the 'Build History' section, which includes a search bar and a list of recent builds. The main content area is titled 'Changes' and displays a list of commit messages for three specific builds.

Build History

Build Number	Timestamp
#7	21-Feb-2022 14:37
#6	14-Feb-2022 22:12
#5	14-Feb-2022 11:45
#4	11-Feb-2022 13:52
#3	10-Feb-2022 10:09
#2	09-Feb-2022 11:44
#1	09-Feb-2022 11:35

Changes

#7 (21-Feb-2022 14:37:12)

1. Initial Branch Commit — Richard / githubweb
2. Linked btTransform with Transform, error with OGLLib — Richard / githubweb
3. fixed opengl linker issue — c1048288 / githubweb
4. Fixed bug unable to find shaders — Richard / githubweb
5. update cmake — David Wong / githubweb
6. Work on rendering items within Game Entity — Richard / githubweb
7. update — Chris Hui / githubweb
8. DX11 replace vulkan & fix some bug — Chris Hui / githubweb
9. Included EvilGenius — Chris Hui / githubweb
10. Included EvilGenius 2 — Chris Hui / githubweb
11. Included EvilGenius 2 — Chris Hui / githubweb
12. Update 15Feb2022 — Chris Hui / githubweb
13. Fixed Graphics Bugs — Richard / githubweb
14. Improved Control and Prefabs — Chris Hui / githubweb
15. Update — Chris Hui / githubweb
16. Change to Game Entity Class — Richard / githubweb
17. Apply FMOD sound — David Wong / githubweb
18. Fixed Cull wall effect — Chris Hui / githubweb
19. Clean up Cull Obstacle - unused class — Chris Hui / githubweb
20. Added Cull Object — Chris Hui / githubweb
21. Simulate Physics — ashbyr51 / githubweb
22. 17/02/2022 — Richard / githubweb
23. 17/02/2022 — Richard / githubweb
24. Renderer and Physics Objects now track same Location — Richard / githubweb
25. Add Player — David Wong / githubweb
26. 18/02/2022 — Richard / githubweb
27. Symmetric Map — Chris Hui / githubweb
28. SceneManager — Richard / githubweb
29. Symmetric Map — Chris Hui / githubweb
30. Symmetric Map & walls — Chris Hui / githubweb
31. Add GameManager — ashbyr51 / githubweb
32. add sound — David Wong / githubweb
33. fmod update — David Wong / githubweb
34. Merged SceneManager — Chris Hui / githubweb
35. update — David Wong / githubweb
36. Merged FMOD — Chris Hui / githubweb
37. Added Collectable Gen — Chris Hui / githubweb
38. Merged MazeGen — Chris Hui / githubweb
39. Merged Every Unity branches except GUI — Chris Hui / githubweb

#6 (14-Feb-2022 22:12:35)

1. testing — Chris Hui / githubweb

#4 (11-Feb-2022 13:52:34)

1. Merged MazeGen — Chris Hui / githubweb

Appendix C – Code Snippets of Integrated middleware into Custom Engine

Physics – Bullet Physics

- Bullet Physics middleware accessed by two provided header files.
- Physics world created with a dynamic's world created in game's constructor
- Physics objects transformed accessed and shared in new game Entity class
- Simulation increments called in existing update loop.

```
#pragma once
```

```
#include "btBulletCollisionCommon.h"  
#include "btBulletDynamicsCommon.h"  
  
#include "../Common/RenderObject.h"  
#include "../Physics/PhysicsObject.h"  
  
#include "../CSC8503/Transform.h"  
  
#include <vector>
```

```
using std::vector;  
using std::string;
```

```
class GameEntity {  
public:  
    GameEntity(string name = "");  
    ~GameEntity();  
  
    //Graphics  
    RenderObject* GetRenderObject() const {  
        return renderObject;  
    }  
  
    void SetRenderObject(RenderObject* newObject) {  
        renderObject = newObject;  
    }  
  
    //Physics  
    btRigidBody* GetRigidBody() const {  
        return rigidBody;  
    }  
  
    void SetRigidBody(btRigidBody* newRigidBody) {  
        rigidBody = newRigidBody;  
    }  
};
```

```
PhysicsTestScene::PhysicsTestScene() {  
    world = new GameWorld();  
    renderer = new GameTechRenderer(*world);  
  
    //Default Broadphase  
    maxProxies = 1024;  
    worldAabbMin = {-100, -100, -100};  
    worldAabbMax = {100, 100, 100};  
    broadphase = new btAxisSweep3(worldAabbMin, worldAabbMax, maxProxies);  
  
    collisionConfiguration = new btDefaultCollisionConfiguration();  
    dispatcher = new btCollisionDispatcher(collisionConfiguration);  
    solver = new btSequentialImpulseConstraintSolver();  
    dynamicsWorld = new btDiscreteDynamicsWorld(dispatcher, broadphase, solver, collisionConfiguration);  
    dynamicsWorld->setGravity(btVector3(0, -10, 0));  
  
    InitAssets();  
    InitCamera();  
    InitScene();  
}
```

```

//ground
ground = new GameEntity("Ground");
ground->GetTransform()
    .SetPosition(Vector3(0, 0, -200))
    .SetScale(Vector3(100, 1, 100))
    .SetOrientation(Quaternion(0, 0, 0, 1));

ground->SetRenderObject(new RenderObject(&ground->GetTransform(), cubeMesh, basicTex, basicShader));

//ground->ConvertToBtTransform();

int groundMass = 0;
ground->GetbtTransform().setOrigin({ 0, 0, -200 });
ground->GetbtTransform().setRotation({ 0, 0, 0, 1 });

btDefaultMotionState* groundMotion = new btDefaultMotionState(ground->GetbtTransform());
btCollisionShape* groundShape = new btBoxShape({ 50, 1, 50 });
btRigidBody::btRigidBodyConstructionInfo groundCI(groundMass, groundMotion, groundShape, btVector3(0, 0, 0));
ground->SetRigidBody(new btRigidBody(groundCI));

world->AddGameObject(ground);
dynamicsWorld->addRigidBody(ground->GetRigidBody());

```

```

void PhysicsTestScene::UpdateGame(float dt) {

    dynamicsWorld->stepSimulation(1 / 60.f, 10);

    world->GetMainCamera()->UpdateCamera(dt);
    UpdateKeys();
    renderer->Render();

    world->UpdatePositions();
}

```

Audio – FMOD

- “common.h” is an FMOD provided header
- ERRCHECK are built in debug log function
- Init system is called in start of main
- The AudioUpdate() function is called within main loop

```
1  #pragma once
2
3  #include "fmod.hpp"
4  #include "common.h"
5  #include "../Common/Window.h"
6  #include <Vector>
7
8  class AudioManager {
9      public:
10         void InitSystem();
11
12         void AudioUpdate(NCL::Window* w);
13     private:
14         FMOD::System      *system;
15         FMOD::Sound       *sound;
16         FMOD::Channel     *channel = 0;
17         FMOD_RESULT       result;
18         void              *extradriverrdata = 0;
19 };
```

```
void AudioManager::InitSystem() {
    result = FMOD::System_Create(&system);
    ERRCHECK(result);

    result = system->init(32, FMOD_INIT_NORMAL, extradriverrdata);
    ERRCHECK(result);

    result = system->createSound(Common_MediaPath("Pickup.wav"), FMOD_DEFAULT, 0, &sound);
    ERRCHECK(result);
}
```

```
void AudioManager::AudioUpdate(NCL::Window* w) {
    Common_Update();

    if (NCL::Window::GetKeyboard()->KeyPressed(NCL::KeyboardKeys::P)) {
        result = system->playSound(sound, 0, false, &channel);
        ERRCHECK(result);
    }

    result = system->update();
    ERRCHECK(result);
}
```

Rendering API – Vulkan

- Finish Technology Implementation Doc

- New Class Vulkan Renderer with Draw Calls
- Initialised in Game Constructor
- Renderer called every frame in Update Game

```

1  #pragma once
2  #include "../Plugins/VulkanRendering/VulkanRenderer.h"
3  #include "../Plugins/VulkanRendering/VulkanShader.h"
4  #include "../Plugins/VulkanRendering/VulkanTexture.h"
5  #include "../Plugins/VulkanRendering/VulkanMesh.h"
6
7  #include "../CSC8503Common/GameWorld.h"
8
9  namespace NCL {
10     class Maths::Vector3;
11     class Maths::Vector4;
12     namespace CSC8503 {
13         class RenderObject;
14
15         class VkTechRenderer : public VulkanRenderer {
16         public:
17             VkTechRenderer(GameWorld& gameWorld);
18             ~VkTechRenderer();
19
20         protected:
21             void    RenderFrame()    override;
22
23             void    InitDefaultRenderPass() override { VulkanRenderer::InitDefaultRenderPass(); };
24             void    InitDefaultDescriptorPool() override { VulkanRenderer::InitDefaultDescriptorPool(); };
25
26
27             //Matrix4 SetupDebugLineMatrix()    const;
28             //Matrix4 SetupDebugStringMatrix()const;
29
30             //VulkanShader* defaultShader;
31
32             GameWorld& gameWorld;
33
34             //void BuildObjectList();
35             //void SortObjectList();
36             //void RenderShadowMap();
37             //void RenderCamera();
38             //void RenderSkybox();
39
40             //void LoadSkybox();
41
42
43             //vector<const RenderObject*> activeObjects;
44
45             //VulkanShader* skyboxShader;
46             //VulkanMesh* skyboxMesh;
47             //VulkanTexture* skyboxTex;
48
49             ///shadow mapping things
50             //VulkanShader* shadowShader;
51             //VulkanTexture* shadowTex;
52             //VulkanTexture* shadowFBO;
53             //Matrix4    shadowMatrix;
54
55             //Vector4    lightColour;
56             //float    lightRadius;

```

```

4  #include "../CSC8503Common/SpringConstraint.h"
5  #include "../../Plugins/OpenGLRendering/OGLMesh.h"
6  #include "../../Plugins/OpenGLRendering/OGLShader.h"
7  #include "../../Plugins/OpenGLRendering/OGLTexture.h"
8  #include "../../Common/TextureLoader.h"
9  #include "../../Plugins/VulkanRendering/VulkanShaderBuilder.h"
10 #include <iostream>
11
12 using namespace NCL;
13 using namespace CSC8503;
14
15 TutorialGame::TutorialGame() {
16     world      = new GameWorld();
17     renderer    = new VkTechRenderer(*world);
18     physics     = new PhysicsSystem(*world);
19
20     onGoing = true;
21     forceMagnitude = 500.0f;
22     useGravity      = true;
23     inSelectionMode = false;
24     layer           = 0;
25     Debug::SetRenderer(renderer);
26
27     bot = new Bot();
28     player = new Player();
29
30     InitialiseAssets();
31 }

```

```

1  #include "VkTechRenderer.h"
2  using namespace NCL;
3  using namespace Rendering;
4  using namespace CSC8503;
5
6  VkTechRenderer::VkTechRenderer(GameWorld& gameworld) : VulkanRenderer(Window::GetWindow()), gameworld(gameworld) {
7
8  }
9
10 VkTechRenderer::~VkTechRenderer() {
11
12 }
13
14 void VkTechRenderer::RenderFrame() {
15
16 }

```

Networking - ENet (Srichand)

- ENet library is integrated into the Game Tech folder by copying the library files into the folder.
- The library functions are used in creating GameServer and GameClient files.


```

1  #include "NetworkedGame.h"
2  #include "NetworkPlayer.h"
3  #include "../CSC8503Common/GameServer.h"
4  #include "../CSC8503Common/GameClient.h"
5
6  #define COLLISION_MSG 30
7
8  struct MessagePacket : public GamePacket {
9      short playerID;
10     short messageID;
11
12     MessagePacket() {
13         type = Message;
14         size = sizeof(short) * 2;
15     }
16 };
17
18 NetworkedGame::NetworkedGame() {
19     thisServer = nullptr;
20     thisClient = nullptr;
21
22     NetworkBase::Initialise();
23     timeToNextPacket = 0.0f;
24     packetsToSnapshot = 0;
25 }
26
27 NetworkedGame::~NetworkedGame() {
28     delete thisServer;
29     delete thisClient;
30 }
31
32 void NetworkedGame::StartAsServer() {
33     thisServer = new GameServer(NetworkBase::GetDefaultPort(), 4);
34
35     thisServer->RegisterPacketHandler(Received_State, this);
36 }

```

```

1  #include "GameServer.h"
2  #include "GameWorld.h"
3  #include <iostream>
4
5  using namespace NCL;
6  using namespace CSC8503;
7
8  GameServer::GameServer(int onPort, int maxClients) {
9      port = onPort;
10     clientMax = maxClients;
11     clientCount = 0;
12     netHandle = nullptr;
13     //threadAlive = false;
14
15     Initialise();
16 }
17
18 GameServer::~GameServer() {
19     Shutdown();
20 }
21
22 void GameServer::Shutdown() {
23     SendGlobalPacket(BasicNetworkMessages::Shutdown);
24
25     //threadAlive = false;
26     //updateThread.join();
27
28     enet_host_destroy(netHandle);
29     netHandle = nullptr;
30 }
31
32 bool GameServer::Initialise() {
33     ENetAddress address;
34     address.host = ENET_HOST_ANY;
35     address.port = port;
36 }

```

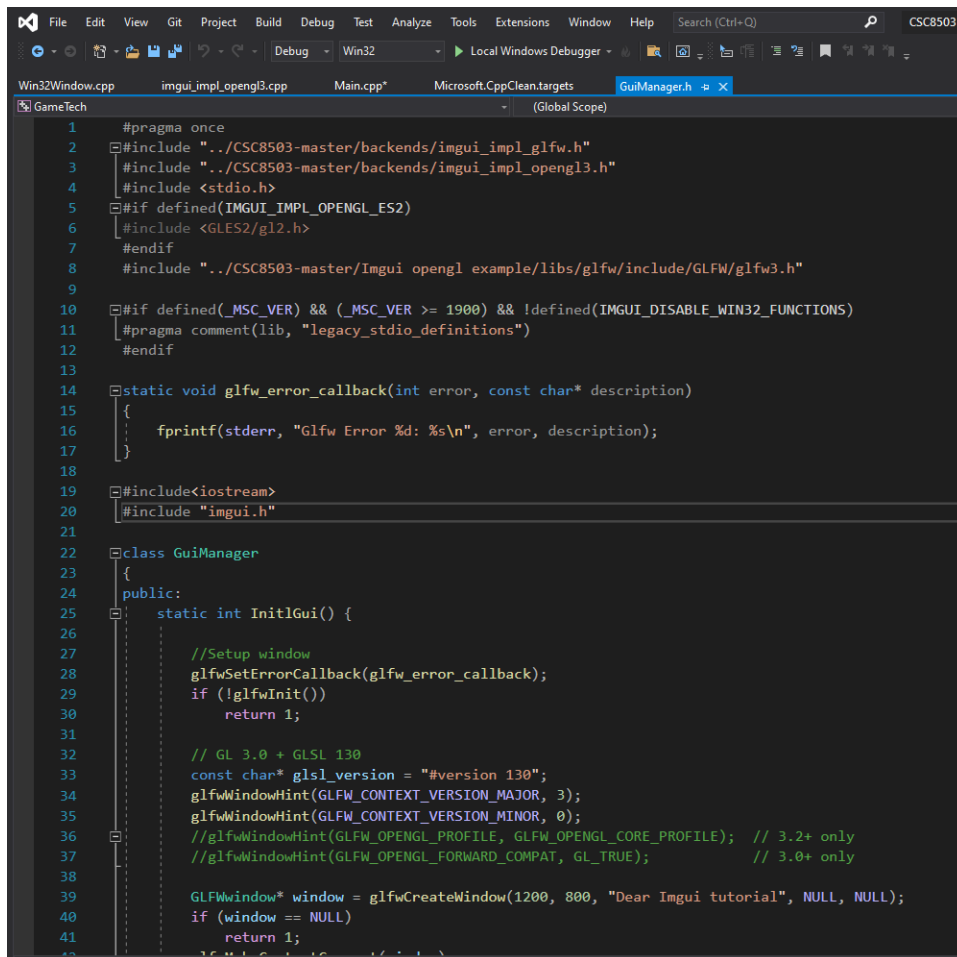
```

1  #include "GameClient.h"
2  #include <iostream>
3  #include <string>
4
5  using namespace NCL;
6  using namespace CSC8503;
7
8  GameClient::GameClient() {
9     netHandle = enet_host_create(nullptr, 1, 1, 0, 0);
10 }
11
12 GameClient::~GameClient() {
13     //threadAlive = false;
14     //updateThread.join();
15     enet_host_destroy(netHandle);
16 }
17
18 bool GameClient::Connect(uint8_t a, uint8_t b, uint8_t c, uint8_t d, int portNum) {
19     ENetAddress address;
20     address.port = portNum;
21
22     address.host = (d << 24) | (c << 16) | (b << 8) | (a);
23
24     netPeer = enet_host_connect(netHandle, &address, 2, 0);
25
26     if (netPeer != nullptr) {
27         //threadAlive = true;
28         //updateThread = std::thread(&GameClient::ThreadedUpdate, this);
29     }
30
31     return netPeer != nullptr;
32 }
33
34 void GameClient::UpdateClient() {
35     if (netHandle == nullptr)
36     {

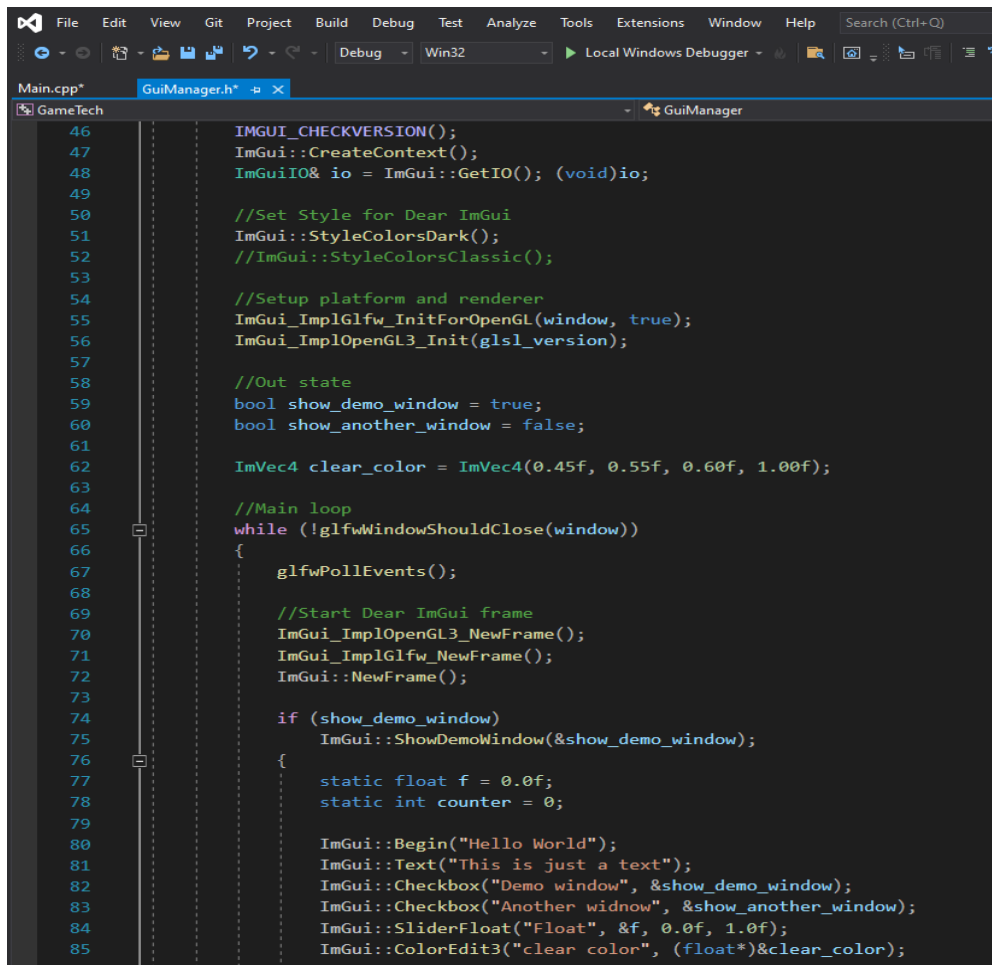
```

GUI- ImGui

- GUI Manager class is created where imgui.h is defined to access the ImGui features
- Integrated a simple ImGui window using glfw and opengl3
- ImGui_impl_glfw and ImGui_impl_opengl3 invokes the backends used for this project
- Gui Manager here creates a function called InitGui which is called in the main function



```
1  #pragma once
2  #include "../CSC8503-master/backends/imgui_impl_glfw.h"
3  #include "../CSC8503-master/backends/imgui_impl_opengl3.h"
4  #include <stdio.h>
5  #if defined(IMGUI_IMPL_OPENGL_ES2)
6  #include <GLES2/gl2.h>
7  #endif
8  #include "../CSC8503-master/ImGui opengl example/libs/glfw/include/GLFW/glfw3.h"
9
10 #if defined(_MSC_VER) && (_MSC_VER >= 1900) && !defined(IMGUI_DISABLE_WIN32_FUNCTIONS)
11 #pragma comment(lib, "legacy_stdio_definitions")
12 #endif
13
14 static void glfw_error_callback(int error, const char* description)
15 {
16     fprintf(stderr, "Glfw Error %d: %s\n", error, description);
17 }
18
19 #include <iostream>
20 #include "imgui.h"
21
22 class GuiManager
23 {
24 public:
25     static int InitGui() {
26
27         //Setup window
28         glfwSetErrorCallback(glfw_error_callback);
29         if (!glfwInit())
30             return 1;
31
32         // GL 3.0 + GLSL 130
33         const char* glsl_version = "#version 130";
34         glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
35         glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 0);
36         //glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE); // 3.2+ only
37         //glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);           // 3.0+ only
38
39         GLFWwindow* window = glfwCreateWindow(1200, 800, "Dear ImGui tutorial", NULL, NULL);
40         if (window == NULL)
41             return 1;
42     }
```



```
46  ImGui_CHECKVERSION();
47  ImGui::CreateContext();
48  ImGuiIO& io = ImGui::GetIO(); (void)io;
49
50  //Set Style for Dear ImGui
51  ImGui::StyleColorsDark();
52  //ImGui::StyleColorsClassic();
53
54  //Setup platform and renderer
55  ImGui_ImplGlfw_InitForOpenGL(window, true);
56  ImGui_ImplOpenGL3_Init(glsl_version);
57
58  //Out state
59  bool show_demo_window = true;
60  bool show_another_window = false;
61
62  ImVec4 clear_color = ImVec4(0.45f, 0.55f, 0.60f, 1.00f);
63
64  //Main loop
65  while (!glfwWindowShouldClose(window))
66  {
67      glfwPollEvents();
68
69      //Start Dear ImGui frame
70      ImGui_ImplOpenGL3_NewFrame();
71      ImGui_ImplGlfw_NewFrame();
72      ImGui::NewFrame();
73
74      if (show_demo_window)
75          ImGui::ShowDemoWindow(&show_demo_window);
76      {
77          static float f = 0.0f;
78          static int counter = 0;
79
80          ImGui::Begin("Hello World");
81          ImGui::Text("This is just a text");
82          ImGui::Checkbox("Demo window", &show_demo_window);
83          ImGui::Checkbox("Another window", &show_another_window);
84          ImGui::SliderFloat("Float", &f, 0.0f, 1.0f);
85          ImGui::ColorEdit3("clear_color", (float*)&clear_color);
86      }
```

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)
Debug Win32 Local Windows Debugger
main.cpp* GuiManager.h*
GameTech GuiManager
97 //Show another window
98
99 if (show_another_window)
100 {
101     ImGui::Begin("Another Window", &show_another_window);
102     ImGui::Text("Hello from another window");
103     if (ImGui::Button("Close me"))
104     {
105         show_another_window = false;
106     }
107     ImGui::End();
108 }
109 //Rendering
110 ImGui::Render();
111 int display_w, display_h;
112 glfwGetFramebufferSize(window, &display_w, &display_h);
113 // glViewport(0, 0, display_w, display_h);
114 //glClearColor(clear_color.x * clear_color.w, clear_color.y * clea
115 // glClear(GL_COLOR_BUFFER_BIT);
116 ImGui_ImplOpenGL3_RenderDrawData(ImGui::GetDrawData());
117
118 glfwSwapBuffers(window);
119 }
120
121
122 system("pause");
123 // Cleanup
124 ImGui_ImplOpenGL3_Shutdown();
125 ImGui_ImplGlfw_Shutdown();
126 ImGui::DestroyContext();
127
128
129 glfwDestroyWindow(window);
130 glfwTerminate();
131
132 return 0;
133 }
134
135 };
136
137
```

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)
Debug Win32 Local Windows Debugger
main.cpp GuiManager.h*
GameTech (Global Scope)
1 #include "../Common/Window.h"
2
3 #include "../CSC8503Common/StateMachine.h"
4 #include "../CSC8503Common/StateTransition.h"
5 #include "../CSC8503Common/State.h"
6
7 #include "../CSC8503Common/NavigationGrid.h"
8
9 #include "TutorialGame.h"
10 #include "../CSC8503Common/PushdownMachine.h"
11 #include "../CSC8503Common/PushdownState.h"
12
13 #include "../GameTech/GuiManager.h"
14
15 using namespace NCL;
16 using namespace CSC8503;
17
18 /*
19  *
20  * The main function should look pretty familiar to you!
21  * We make a window, and then go into a while loop that repeatedly
22  * runs our 'game' until we press escape. Instead of making a 'renderer'
23  * and updating it, we instead make a whole game, and repeatedly update that,
24  * instead.
25  *
26  * This time, we've added some extra functionality to the window class - we can
27  * hide or show the
28  */
29 */
30
31 Window* w;
32 TutorialGame* g;
33
34 int reset;
35
36 enum MenuOptions
37 {
38     l1,
39     l2,
40     quit,
41 };

```

```

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) CSC8503
Debug Win32 Local Windows Debugger
main.cpp GuiManager.h*
GameTech (Global Scope)
115
116 };
117
118
119 int main() {
120     Window* w = Window::CreateGameWindow("CSC8503 Game technology!", 1280, 720);
121
122     if (!w->HasInitialised()) {
123         return -1;
124     }
125     srand(time(0));
126     w->ShowOSPointer(false);
127     w->LockMouseToWindow(true);
128
129     GuiManager* g1;
130
131     g1->InitlGui();
132
133     TutorialGame* g = new TutorialGame();
134     w->GetTimer()->GetTimeDeltaSeconds(); //Clear the timer so we don't get a target first dt!
135     while (w->UpdateWindow() && !Window::GetKeyboard()->KeyDown(KeyboardKeys::ESCAPE)) {
136         float dt = w->GetTimer()->GetTimeDeltaSeconds();
137         if (dt > 0.1f) {
138             std::cout << "Skipping large time delta" << std::endl;
139             continue; //must have hit a breakpoint or something to have a 1 second frame time!
140         }
141         if (Window::GetKeyboard()->KeyPressed(KeyboardKeys::PRIOR)) {
142             w->ShowConsole(true);
143         }
144         if (Window::GetKeyboard()->KeyPressed(KeyboardKeys::NEXT)) {
145             w->ShowConsole(false);
146         }
147
148         if (Window::GetKeyboard()->KeyPressed(KeyboardKeys::T)) {
149             w->SetWindowPosition(0, 0);
150         }
151
152         w->SetTitle("Gametech frame time:" + std::to_string(1000.0f * dt));
153
154         g->UpdateGame(dt);
155     }

```