Competencies

Cameron Soal – b8010556

|  |  |  |
| --- | --- | --- |
| Competency | Example of Usage | |
| Tutorial 1 – Text Rendering and basic algorithms. | See PlayerUI.h, lines 16, 32, and 37.  See NodeTypes.h, lines 52>74.  See ItemShopMode.h, lines 36, 73 and 74.  For basic algorithms, see entire project, various examples present. | See PlayerUI.cpp, MenuText objects are defined at various points.  See NodeTypes.cpp, lines 100>137.  For basic algorithms, see entire project, various examples present. |
| Tutorial 2 – Classes, Objects and Sprite Rendering. | For Classes/Objects, see entire project. Various examples used.  For Sprite Rendering, see TextureCache.h and Sprite.h | For Classes/Objects, see entire project. Various examples used.  For Sprite Rendering, see TextureCache.cpp and Sprite.cpp |
| Tutorial 3 – Dynamic Files and Version Control. | See entire project. #pragma once and ifndef used throughout. | See entire project. #pragma once and ifndef used throughout. |
| Tutorial 4 – Generic Reusable classes, Key techniques. | See entire project. GameObject inherited by all objects on screen and reused.  See PlayMode.h, line 53. Vector used containing GameObject pointers to store all objects in 1 place.  Player.h, line 19, uses a vector container to hold generic weapon pointer.  Pointer and reference types used throughout the project.  Singleton.h reused multiple times to delare objects as “Singleton” such as PlayMode.h, MainGame.h, ItemShopMode.h  Polymorphism used throughout project. | See Singleton.h, line 24, overloading = operator |
| Tutorial 5 – Test plans, Interfaces, Collision Detection | For Test plans, see tutorial documentation and Unnamed Space game UML.  For Collision Detection, see Collider2D.h | For Collision Detection, see Collider2D.cpp.  Lambdas used for button events, see ItemShopMode.cpp, lines 235 and 470 |
| Tutorial 7/8 – Using GiT, DX11 Techniques and Sprites | GitHub used for repos, evident by inclusion of a .gitignore file in all SFML content and the DX11 Game.  Sprites logic found in Sprite.h/.cpp  Texture logic found in TextureCache.h/.cpp  Transparency used in textures by exporting .dds files in RGBA8 format.  Graphics pipeline used in D3D.h/.cpp  winOS message handling implemented in main.cpp, line 13.  Frame timing implemented in main.cpp, lines 49>63. | All assets loaded into texture cache in MainGame.cpp, lines 70>112. |
| Tutorial 9/10 – Gamepad and Mouse and Keyboard input support. | See Input.h/.cpp and GamePadInput.h/.cpp for implementation. | Inputs handled in Player.cpp, lines 103>163 and lines 184>196. |
| Tutorial 11 – Polymorphic User Interface, Data structures for performance and memory optimisation. | See PlayerUI.h/.cpp and ItemShopMode.h/.cpp.  Inner structs and classes used in order to create re-usable data structures for UI components.  Classes release memory where necessary, see ItemShopMode.cpp, lines 152>160 and D3D.cpp lines 53>94 as an example. Some classes use smart pointers to automatically clean up after themselves. See PlayMode.h, line 53 as an example of using a smart pointer to automatically deallocate memory after all instances are destroyed. | Example, ItemShopMode.h lines 14>44 defines a “Buyable” struct, which is used in ItemShopMode.cpp lines 201>204 to create items in the item shop for the player to buy. |