3D Modeling Assignment 2 : Constructing and displaying an airport model.

Ryan Hood

P1617694X

Introduction.

The goal of this assignment was to be able to create a functional model loader and load in multiple different models to construct an airport. We needed to use a mixture of both C++ and openGL to build the model loader.

The model loader needed to be able to read a number of different line within a (.obj) file and save all of the data from it. It first needed to read the v line then if there were any the vn and finally the vt lines. Vectors for the data needed to be created to store the data so that they could be drawn later.

After the model was created the a texture needed to be associated with it to give the scene more depth. So thus a texture loader was required, using the provided C++ and header files I slotted the texture loader into my existing code and it worked perfectly.

With that I had all of the tools I needed to construct the scene, I figured out how to position the models within the scene and quickly added the corresponding functions into my model class.

However, due to a lack of time after having an embarrassing number of issues with the actual model loading from an (.obj) file itself; I was unable to implement a system where the scene could be easily loaded from an external file and had to resort to hard coding the scene into the program.

What Went Well.

After I was able to implement a system wherein the model could be loaded into the scene from an (.obj) file I was rather quick at implementing the following systems; ones like loading the textures from the file, positioning the models and the interaction the classes had with each other.

What Was Learnt.

I now have a firm grasp on how to use openGl and I can now confidently load models and textures from an external file.

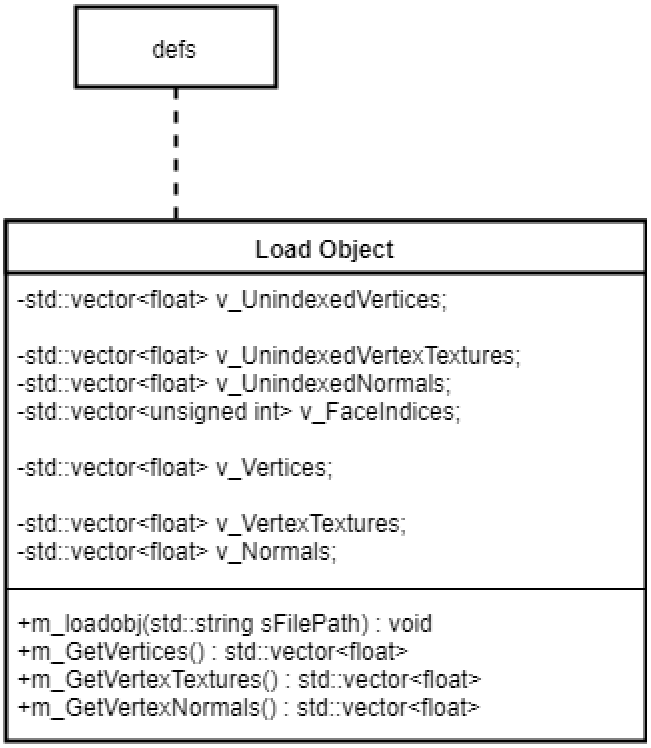
I have learnt and become more adept at using the 3D modeling software Maya, a skill which will be pretty useful in the future no doubt.

Improvements.

If I were to attempt this assignment again with fresh eyes, I would most importantly add in a more fluid system for moving the camera, ideally one which centers on mouse movement to control what the user is looking at.

The next improvement I would make would be to include a more diverse selection of models in the scene, at first I was rather ambitious with the number of models I wanted to include in the scene, (as shown within my reference images folder); but upon my first stumble I lost a lot of time attempting to fix my issues, and thus was unable to create my grand scene.

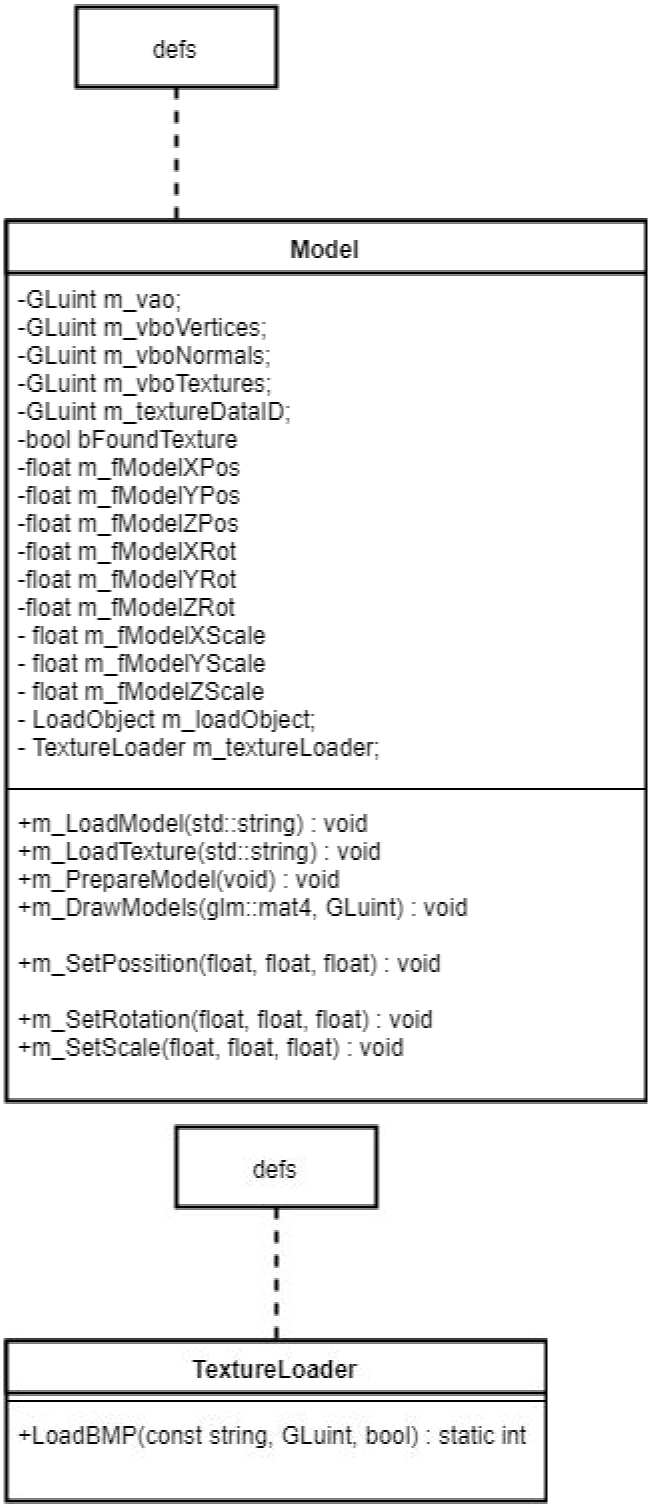
Class Diagrams :



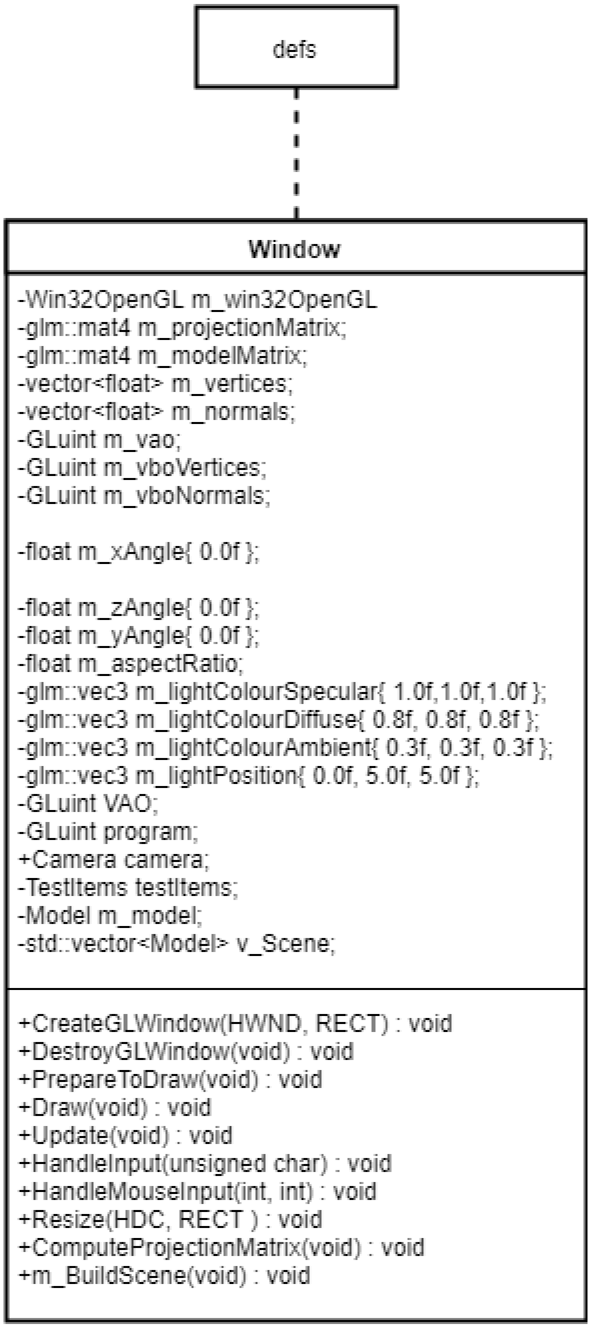
```
-std::vector<float> v_UnindexedVertices;

-std::vector<float> v_UnindexedVertexTextures;
-std::vector<float> v_UnindexedNormals;
-std::vector<unsigned int> v_FaceIndices;

-std::vector<float> v_Vertices;

-std::vector<float> v_VertexTextures;
-std::vector<float> v_Normals;
```

```
+m_loadobj(std::string sFilePath) : void
+m_GetVertices() : std::vector<float>
+m_GetVertexTextures() : std::vector<float>
+m_GetVertexNormals() : std::vector<float>
```

3D Modeling Assignment 2 : Constructing and displaying an airport model.
Ryan Hood
P1617694X


defs
Model
-GLuint m_vao;
-GLuint m_vboVertices;
-GLuint m_vboNormals;
-GLuint m_vboTextures;
-GLuint m_textureDataID;
-bool bFoundTexture
-float m_fModelXPos
-float m_fModelYPos
-float m_fModelZPos
-float m_fModelXRot
-float m_fModelYRot
-float m_fModelZRot
- float m_fModelXScale
- float m_fModelYScale
- float m_fModelZScale
- LoadObject m_loadObject;
- TextureLoader m_textureLoader;
+m_LoadModel(std::string) : void
+m_LoadTexture(std::string) : void
+m_PrepareModel(void) : void
+m_DrawModels(glm::mat4, GLuint) : void
+m_SetPossition(float, float, float) : void
+m_SetRotation(float, float, float) : void
+m_SetScale(float, float, float) : void
defs
TextureLoader
+LoadBMP(const string, GLuint, bool) : static int

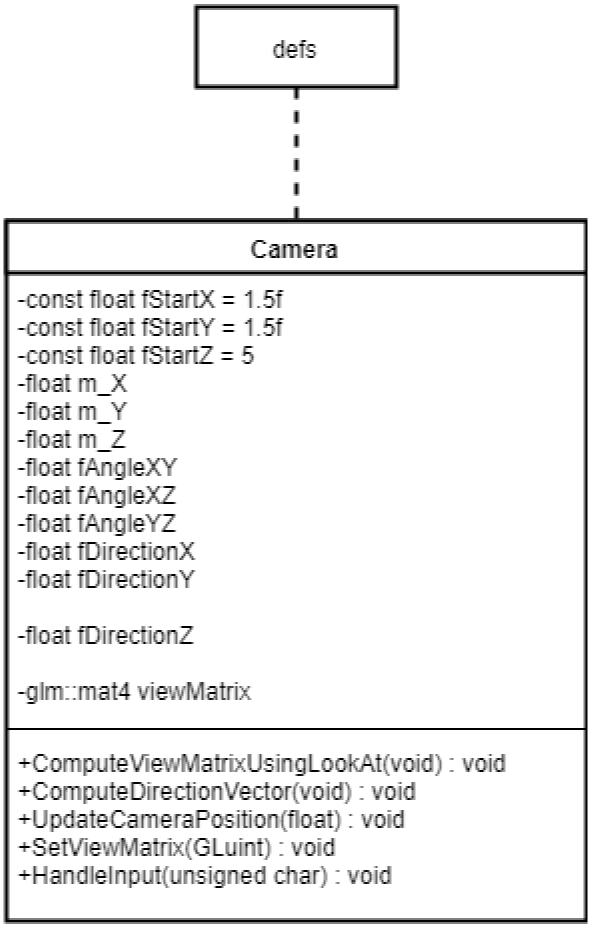
3D Modeling Assignment 2 : Constructing and displaying an airport model.
Ryan Hood
P1617694X



```
-Win32OpenGL m_win32OpenGL
-glm::mat4 m_projectionMatrix;
-glm::mat4 m_modelMatrix;
-vector<float> m_vertices;
-vector<float> m_normals;
-GLuint m_vao;
-GLuint m_vboVertices;
-GLuint m_vboNormals;

-float m_xAngle{ 0.0f };

-float m_zAngle{ 0.0f };
-float m_yAngle{ 0.0f };
-float m_aspectRatio;
-glm::vec3 m_lightColourSpecular{ 1.0f,1.0f,1.0f };
-glm::vec3 m_lightColourDiffuse{ 0.8f, 0.8f, 0.8f };
-glm::vec3 m_lightColourAmbient{ 0.3f, 0.3f, 0.3f };
-glm::vec3 m_lightPosition{ 0.0f, 5.0f, 5.0f };
-GLuint VAO;
-GLuint program;
+Camera camera;
-TestItems testItems;
-Model m_model;
-std::vector<Model> v_Scene;
```

```
+CreateGLWindow(HWND, RECT) : void
+DestroyGLWindow(void) : void
+PrepareToDraw(void) : void
+Draw(void) : void
+Update(void) : void
+HandleInput(unsigned char) : void
+HandleMouseInput(int, int) : void
+Resize(HDC, RECT ) : void
+ComputeProjectionMatrix(void) : void
+m_BuildScene(void) : void
```

3D Modeling Assignment 2 : Constructing and displaying an airport model.
Ryan Hood
P1617694X



defs

Camera

-const float fStartX = 1.5f
-const float fStartY = 1.5f
-const float fStartZ = 5
-float m_X
-float m_Y
-float m_Z
-float fAngleXY
-float fAngleXZ
-float fAngleYZ
-float fDirectionX
-float fDirectionY

-float fDirectionZ

-glm::mat4 viewMatrix

+ComputeViewMatrixUsingLookAt(void) : void
+ComputeDirectionVector(void) : void
+UpdateCameraPosition(float) : void
+SetViewMatrix(GLuint) : void
+HandleInput(unsigned char) : void

3D Modeling Assignment 2 : Constructing and displaying an airport model.
Ryan Hood
P1617694X