WM9M2-15 Computer Graphics

Iroegbu Charles
5588649

University of Warwick

December 16, 2024

A 3D game developed in C++ and DirectX 11.

1 Introduction

The repository link is https://github.com/charlescookey/ComputerGraphicsStart

The project was to create a 3D game using C++ and DirectX 11. The goal of this project was to implement advanced graphics techniques taught in class. The game is a basic game where the player controls a character and hunts for Tyrannosaurs. The game has a lit 3D level, where meshes are rendered in. Data driven loading is employed as the position of characters and objects are read from custom file formats. The game includes a third person camera which is controlled using the keyboard and mouse. The game includes Animated and Static meshes, which responds to collision with other meshes. The game also employs advanced graphic techniques such and Shadow mapping and deferred shading.

The game involves finding the trex and using "F" to shoot, and after multiple shots, the Trex dies.

2 Method

Describe your method here. Be concise, accurate and scientific. Don't talk about the process you went through. Do describe the technical aspects of your prototype, and why you made certain decisions. Include references to Figures like this: Figure 8 shows a car rendered with path tracing. Make sure you always refer to each figure from the text. But only add them if they contribute something to watch you want to describe. If you need to include maths, this is what you need:

2.1 Textured and Lit Levels

Texturing is adding surface details and color to a 3D model. The textured are mapped using UV coordinates, this is mapped into the XY coordinates. Sampling is used to query the value at a particular UV coordinate and get the corresponding texture value.

As shown here https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Shaders/TextureMeshPixel.txt#L15

float4 colour = tex.Sample(samplerLinear, input.TexCoords);

The color is gotten for a perticular pixel by using Linear Smapling to get the texture value from an input Texture Image at a UV coordinate.

2.2 Mesh Rendering

In the game, different types of meshes are rendered, basic geometries like Cubes, Sphere and rendered by specifying the vertices and the indices to render. https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Model.h#L96 Static and Animated Meshes are also rendered, but the vertices and indices are read from custom file formats. All meshes are basically a collection of vertices, edges and faces that define the 3D object.

```
1 0, 1.0f, 0
2 -1.0f, -1.0f, 0
3 1.0f, -1.0f, 0
```

For example, above is the vertices for a triangle.

2.3 Data-Driven Level Loading

The meshes are loaded from a custom file format ".gem". This contains all infpormation neccesary

Figure 1: *This is an example of a texture map that would be applied to the surface of a 3d model.*

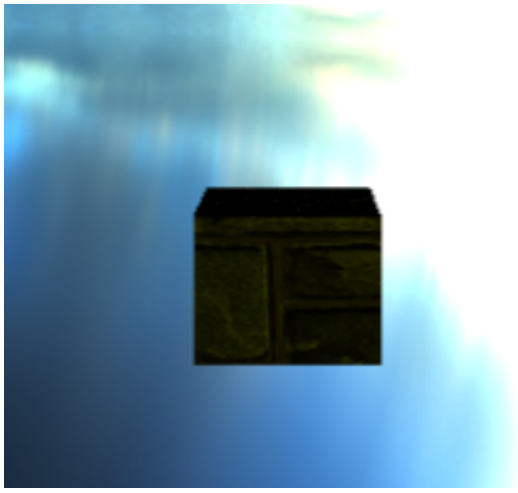

Figure 2: *This is the texture map applied to a 3d model.*

Figure 3: *Images showing a tree with and without alpha testing.*

to load the 3D image. https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Model.h#L43

The GEMLoader is used to read the vertices,indices,animations from the gem models into the code to render the meshes. Details like position of the object in the scene are read from textfiles. https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/InstanceMesh.h#L94

Here we are reading the position for the trees to be rendered using instancing to avoid multiple calls

2.4 Opaque meshes

Sapling a texture gives us a float4 which is basically an array of size 4, where the first three are the RGB values, and the fourth is the Alpha Value, This specifies the opcaity of elements. Using a scale of 0-1, where 1 is fully opaque and 0 is completely transparent

2.5 alpha-testing

As discussed in the previos section, the alpha value specifies the opacity, we set a threshold, below a certain value, the shader should not render the tecture for the specifies UV coordinate. https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Shaders/TextureMeshPixel.txt#L16

Here the value is set at 0.5, so if the alpha value is below this it won't be rendered.

2.6 Vertex Shader Animation

the vertex shader process the vertices before rasterization and applies some transformation tp produces required output. Animating vertices directly in the shader can create dynamic effects, such as waving tree branches.

https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Shaders/StaticMeshVertexAnimation.txt#L31

Here the vertex shader creates a waving animation by using sine wave transformations. The oscillation

Figure 4: *vertex Animation.*

value is added to the Y value to make it move up and down

2.7 Camera Movement

A camera class was created to move the camera based on the input of the user. WASD is used for movement and mouse input for orientation. The view matrix is updated each frame based on position of the Charcater which os used as the To (Target) and the Camera which is used as the from. https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Camera.h#L11

2.8 Animated Game Characters

https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Character.h#L5 The Character class contains animation states which are used to control which animations are played based on certain input. If the Character is Shot, the animation state switches to a React animation, and of the Character is dead, a death animation is played

2.9 Collision Detection and Response

The AABB (Axis-Aligned Bounding Boxes) is employed for Models, This fits a box around the model. The box is created by running through all the vertices to find the minimum and maximum vales on the x,y and z axis. The Ray is used for detecting collision when Character shoots by employing the RayAABB collision detection.
https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Collision.h#L4

2.10 Deferred Shading

Theory: Deferred shading separates geometry rendering and lighting into two passes, improving performance by reducing redundant lighting calculations.

2.11 Shadow Mapping

Shadow mapping involves rendering the scene from the light's perspective to create a depth map. Shadows are determined by comparing depths from the camera and light perspectives.

This is done in two passes, first map is rendering the world from teh view of the light source. This would be rendered to a texture and stored as the depth map

Figure 5: *Depth Map.*

The second pass, for each sample we project and compare depths. If it is farther then we use the depth map
https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Shaders/ShadowPixel.txt#L1

2.12 Normal Mapping

Normal mapping is a form of bump mapping that is used to produce a 3D effect using just two 2D textures. This is done by replacing the vertex normals with texture based normals stored in a normal map

2.13 Effects

2.13.1 Fire

https://github.com/charlescookey/ComputerGraphicsStart/blob/eee75a1e1daaa85314e50882e570e2c4c7c6ecde/Shaders/FirePixel.txt#L1

2.13.2 Sky Dome

A skybox is a large sphere that surrounds the players view point to simulate a distant environment. We apply the texture of a sky.

3 Evaluation

The game maintained a steady framerate of over 60 FPS in most areas but dropped during scenes with heavy particle effects or multiple lights.

Limitations

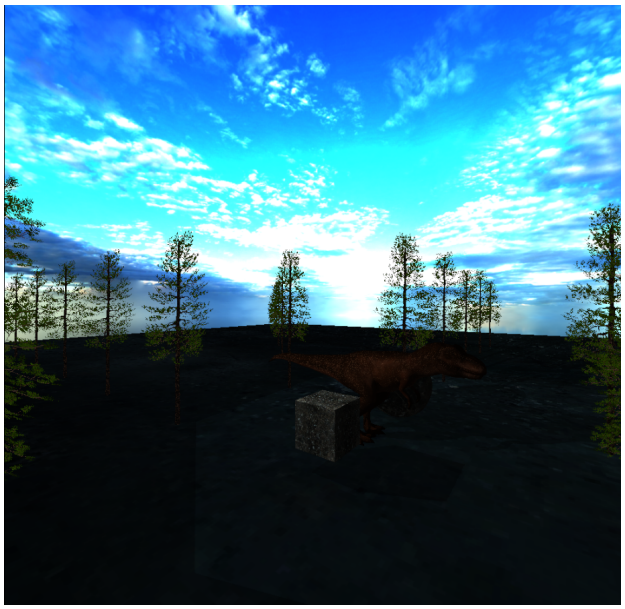

Figure 6: *Shadows.*

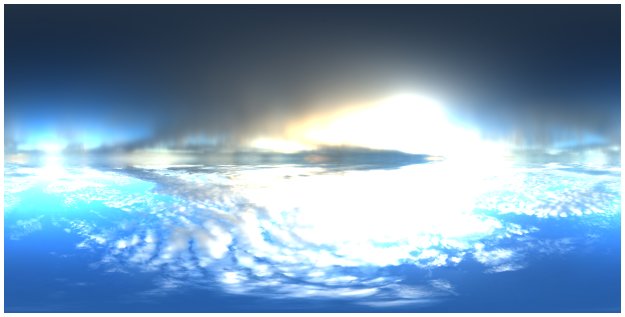

Figure 7: *The Sky Texture.*

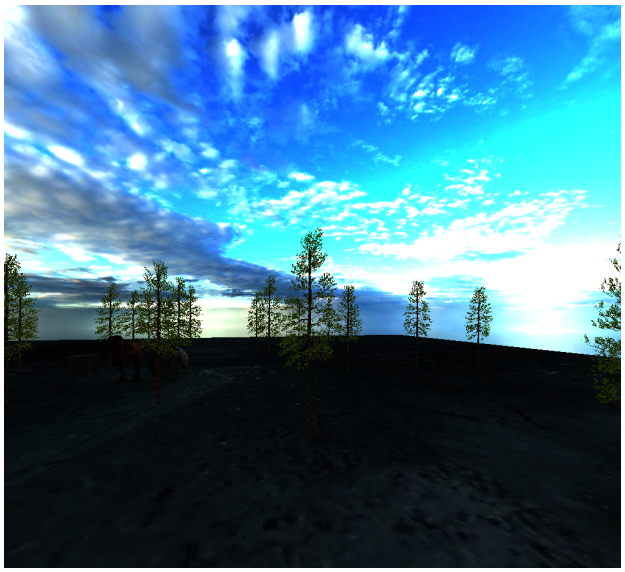

Figure 8: *The Sky Texture mapped to a sphere for the Sky Dome.*

The collision detection system was limited to AABB, which caused inaccuracies with object with complicated figures.

Font didn't work work. More lights in scene

Conclusion

Evaluation goes here

4 Conclusion

This project successfully implemented several advanced graphics techniques . The process provided invaluable experience with DirectX 11 and shader programming. While some features were limited, the overall outcome met the project's goals.

Add a paragraph to say what you did.