January 13, 2019

Oliver Mills

Q12339911

Advanced games programming ae2

cgp600

Contents

[Mechanics 2](#_Toc535343550)

[Movement 2](#_Toc535343551)

[Enemies 2](#_Toc535343552)

[Graphics 2](#_Toc535343553)

[Textures 2](#_Toc535343554)

[Model and texture mangers 2](#_Toc535343555)

[Lighting 2](#_Toc535343556)

[Text 2](#_Toc535343557)

[Particles 2](#_Toc535343558)

[Design patterns 3](#_Toc535343559)

[Classes 3](#_Toc535343560)

[Game loop and timer 3](#_Toc535343561)

[Singleton 3](#_Toc535343562)

[Logic 3](#_Toc535343563)

[Loading a level with a text file 3](#_Toc535343564)

[Collision 3](#_Toc535343565)

[Testing 4](#_Toc535343566)

[Black Box Testing 4](#_Toc535343567)

[Conclusion 5](#_Toc535343568)

[References 5](#_Toc535343569)

[Appendix 6](#_Toc535343570)

[Figure 1 6](#_Toc535343571)

[Asset List 7](#_Toc535343572)

Mechanics

Movement

Simple movement with WASD and space, for activating the jetpack and mouse input for looking around and activating rainbow particles. Only movement with WASD was planned. I implemented the others to better show off features.

Enemies

Enemies ended up much simpler than I had originally planned due to not implementing my array based BSP system the enemies ended up with only simple chase features and being able to collide with objects in the scene.

Graphics

Textures

I developed a system to allow models to have either one or two textures where the second texture will appear on top of the primary texture. Which is what was planned, this works by multiplying the two textures together.

Model and texture mangers

This worked together with the texture manager which would store all the textures to prevent loading duplicates, which makes use of a map to store both a texture and a sampler using the file name as a key. This was more complicated when it came to handle two textures on the same model. I did this by passing in two file names and concatenating them together to form the key for the map, which involved using the string stream to concatenate them.

The model manager was much easier to implement as all that needed to be stored was the objFileModel class and the file name it was the key.

Lighting

I implemented diffuse, ambient and specular lighting following phong principles to calculate colours based on the pixel shader. The formula for phong lighting. *Dot(N, L) = |N||L|cos(α)* (Feinstein, 2013)

Text

I use text to display the FPS using a custom font I created which is intended for use with alpha blending which I also implemented. This is simply implemented by changing the blend state for when then the text is drawn.

Particles

Each particle generator stores a pool of particles ready to be used and depending on inputs provided will result in displaying different types of particles. When the player jumps grey dust like particles will rise upwards, while if the player left clicks rainbow particles will spawn. All of the particles also implement alpha blending so they blend into the scene better.

Design patterns

Classes

The previous class design that I did for AE1 ended up being completely ignored, as once I had done most of the tutorials the structure of my project was vastly different from what I had intended prior to learning Dx11 and I no longer felt this design would work without having to entirely redo everything to allow for it to partially match this naive design.

Game loop and timer

I created a time class to allow the game to run consistently even if frame rates are inconsistent. I found out how to do this by following the section in a DX11 book. (Luna, n.d.) and in combination with a game loop which allows the timer to tick for each frame.

Singleton

Both the model and texture manager follow the singleton design pattern along with the game timer class. To allow them all to be accessible from anywhere, as the timer is required for delta time to make features like movement speed consistent regardless of framerate. While the model and texture managers need to be accessible to every game object as not to waste memory by loading duplicates.

Logic

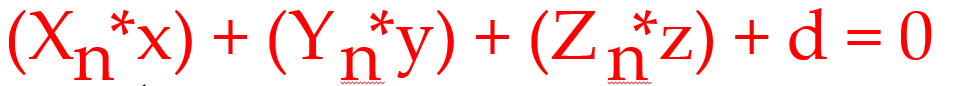
Loading a level with a text file

A simple level creation system where strings are passed into a vector and looped through one character at a time, are then compared against cases of a switch which determines which model should be loaded and sets its location based on its position in the text file.

Collision

Collision changed a lot from how I intended to do it originally, as after looking at the assignment’s marking scheme I realised a lot of the marks were locked behind implementing triangle collision. In attempting to implement it I spent a solid four days struggling to programme it, only to end up with it working inconsistently. Whilst arguably it looks like sphere collision, it is not, as sphere collision is used to determine whether triangle collision should be checked. 

Sphere collision equation (Alassad, 2018, A)

  
The plane equation (Alassad, 2018, B)

The inconsistences are when it comes to two low triangle count models collide, for example two cubes as they will just phase right through each other. While when using one higher triangle count model like the robot they will collide correctly and stop.

Testing

Along with this test plan I also performed ad-hoc testing during development but due to vast number of tests and issues I had during development It would be impractical to cover them, it would have also drastically reduced development progress to have to swap gears and fill in tables about the issues I was having.

Black Box Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Feature | Test | Predicted Outcome | Actual Outcome | Actions Taken |
| 01 | Movement | Press WASD | Movement in four directions based on facing direction | Movement in four directions based on facing direction which take collision into account |  |
| 02 | Sphere Collision | Try to walk into a wall | Gets stopped by wall | Gets stopped by wall |  |
| 03 | Triangle Collision | Try to walk into a wall | Gets stopped by wall | Gets stopped by wall |  |
| 04 | Lighting | See if the ambient light visible | To be able to objects in the scene | Able to objects in the scene |  |
| A |  | See if the directional light visible | To be able to see an addition lighting colour coming from one direction | Able to see an addition lighting colour coming from one direction |  |
| B |  | See if specular highlights visible | able to see an addition lighting highlight coming from one direction | See figure 1 | Checked which colours and how they were being multiplied to affect the final colour.( See figure 1) |
| 05 | Enemies | Stand still,  Do they walk towards the player | They move towards the player and collide with them | They move towards the player and collide with them |  |
| 06 | textures | Run game with objects with a texture | See textures on models | Able to see textures on models |  |
| A | Two textures | Run the game with an object that has two textures | See two textures exclusively on the models that should only have two textures | The second texture appeared on all of the models regardless of if they should have them or not. | To fix this I had to create a second shader which would be used if a model had two textures. this allowed all models to only display the correct amount of textures |
| 07 | Level loading | Run the game with a setup text document with appropriate values | To see a level in which the specified objects are spawned accurate to the text file | Level appears as expected |  |
| 08 | jumping | Press space | Player jumps and particles spawn | Player jumps and particles spawn |  |

Conclusion

I had lots of issues when trying to implement triangle collision, where objects would just phase though it each other and the player could walk through walls. Most of the issues continued for objects with low triangle counts as I couldn’t resolve it. Although the collisions ended up working fine for more complex shapes.

I had several issues with programming shaders and lighting as I struggled to understand how to properly pass values in especially when it came to implementing the more complex lighting systems. And typos plagued shader development.

If I were to develop another Dx11 game, I would be able to have a much better understanding of how to appropriately design classes and the project layout as my initial design did not take into account how DX11 functions due to a lack of understanding, now I have a better understanding I could structure the design to take into account scene management.

References

ALASSAD, P 2018, A, Bounding Sphere Collision Detection, *lecture delivered to Advanced Games Programing Level 6*

ALASSAD, P 2018, B, Planes and the Plane Equation, *lecture delivered to Advanced Games Programing Level 6*

LUNA, F.D., n.d. Introduction to 3d Game Programming with Directx® 11. Dulles, VA: Mercury Learning and Information

FEINSTEIN, D., 2013. HLSL Development Cookbook: Implement Stunning 3D Rendering Techniques Using the Power of HLSL and DirectX 11. Birmingham: Packt Pub.

Appendix

Figure 1

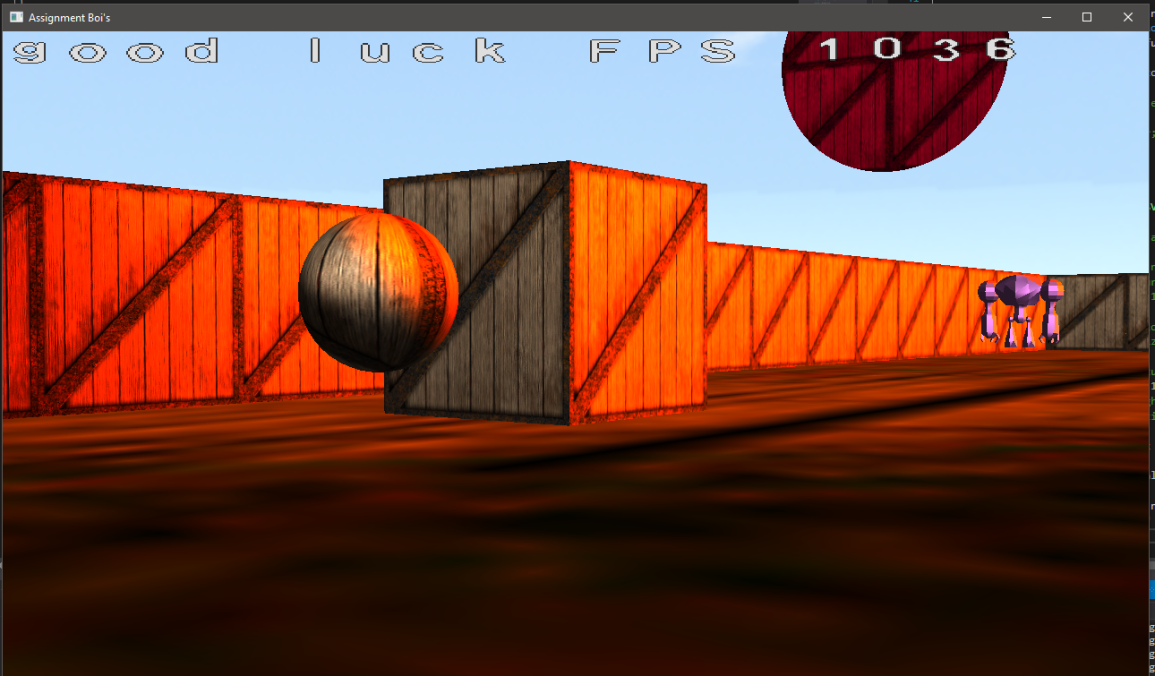
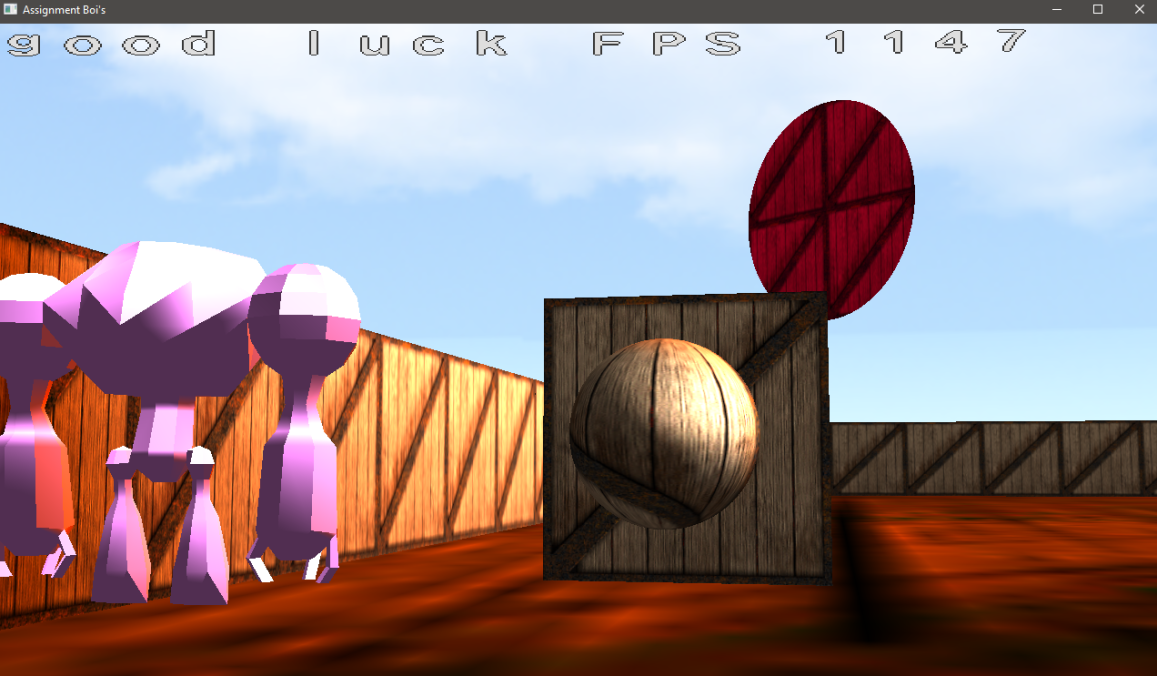


Figure 1 – before fixed. The colours were being multiplied incorrectly, direction light colour was being used for the specular highlight causing a further red hue. To fix this I created a local colour value set to pure white, once that was being using for the multiplication the colour was being displayed as desired.



Asset List

|  |  |  |  |
| --- | --- | --- | --- |
| Asset Name | Asset Type | Source | License |
| DaylightSkybox | Skybox | https://opengameart.org/content/sky-box-sunny-day | CC by 3.0 |
| giant robot | Model | https://clara.io/view/7c7d4477-155b-4690-9dc9-69b49af2748f# | CC BY-NC 3.0 |