January 13, 2019

Oliver Mills

Q12339911

Advanced games programming ae2

cgp600

Contents

[Mechanics 1](#_Toc535321703)

[Graphics 1](#_Toc535321704)

[Design patterns 2](#_Toc535321705)

[Logic 2](#_Toc535321706)

[Design Changes 2](#_Toc535321707)

[New Things that weren’t planned 2](#_Toc535321708)

[Things that changed 2](#_Toc535321709)

[Things That weren’t implemented 3](#_Toc535321710)

[Things that stayed the same 3](#_Toc535321711)

[Testing 3](#_Toc535321712)

[Conclusion 4](#_Toc535321713)

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 - they terrible

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 – multi

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.

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 :’( #hlsl cookbook pg15 – phong equation

I implemented diffuse, ambient and specular lighting following phong principles to calculate colours based on the pixel shader. (insert PHONG here )

Text – alpha blending

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

OOD

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.

Double buffering – we did do that right?

Game loop

Time ? singleton – DX11 handbook page 200 or something.

Texture and model manager

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 - tri and sphere no AABB #insert plane 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 three-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.

Testing

Black Box Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Feature | Test | Predicted Outcome | Actual Outcome | Actions Taken |
|  | Movement | Press WASD | Movement in four directions based on facing direction | Movement in four directions based on facing direction which take collision into account |  |
|  | Sphere Collision | Try to walk into a wall | Gets stopped by wall | Gets stopped by wall |  |
|  | Triangle Collision |  |  |  |  |
|  | Lighting |  |  |  |  |
|  | Enemies |  |  |  |  |
|  | textures |  |  |  |  |
|  | Level loading |  |  |  |  |
|  | jumping |  |  |  |  |

White box Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function Name | Syntax & form | logic | outcome | Actions to be taken |
|  | Is it good? does it need to be clearer? | Does it provide the right answers | Does it need improving? | What to improve? |
|  |  |  |  |  |
|  |  |  |  |  |

Conclusion

Discussion on problems and how they were solved