100752341 - **Kainat Rashid (Artist)**
100661019 - **William Tu (Artist)**
100782680 - **Natasha Bowles (Artist)**
—----------
= **10**

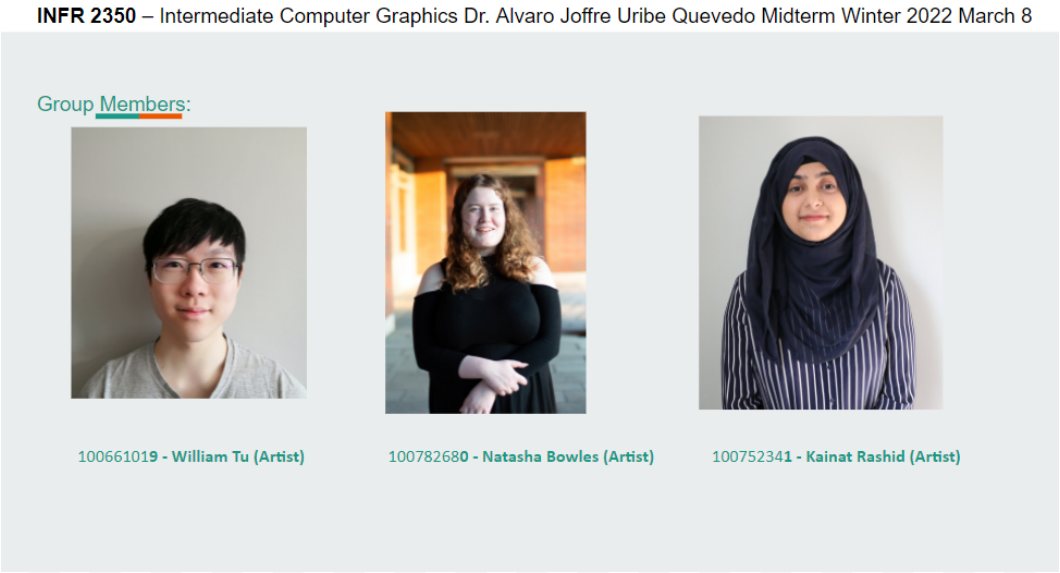
Team Member Contributions

Natasha Bowles - Natasha worked on the document, specifically the lighting and storyboards. She also generally helped with different explanations for some of the answers. Natasha will also be submitting and creating the repo.

William Tu - Worked on parts of the explanation of concepts and implementation. Also made textures for the game

Kainat Rashid- Worked on the document and on the explanation of concepts. Worked on the art assets to help contribute to the storyboard. Also worked on the implementation portion and specifically worked on the Shader section.

The tools used were Blender for lighting, Krita and Photoshop for textures, and Clip studio for the story board.



Game chosen:

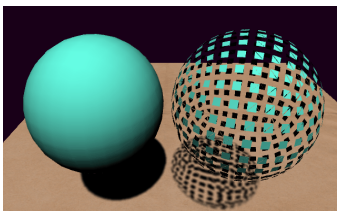
The game we chose was Mario Bros. We wanted to do the first level of the game where Mario has to get through enemies to reach the end of the level. Mario is a movable character with WASD keys. Goomba is the obstacle/enemy that one would try to beat. The lose condition would happen if Mario

gets touched by a Goomba more than once. The winning condition would be to reach the end of the level and make contact with the flag pole.

Explanation of Concepts

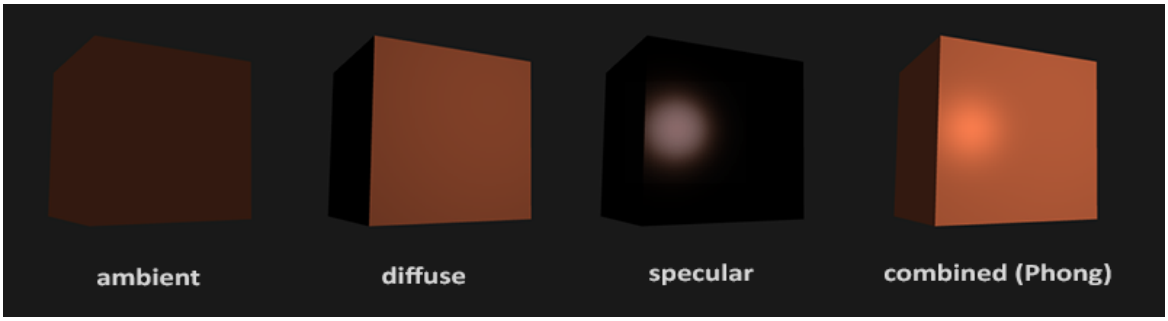
1. **Select the playable character and use it to explain the graphics pipeline stages associated with Vertex shader, Geometry Shader, and fragment shader.**

- The vertex shader receives data from the vertex stream and places it in the output vertex stream one at a time.
- In our Mario Bros level the vertex shader would help map out the world coordinates, colour values, normal values, texture coordinates.
- This stage of the pipeline is vital to plan out the world and you get to start implementing the colour scheme and textures of the world. Mario would have been able to move according to the world coordinates and interact with obstacles.
- The geometry shade is an optional stage in the graphics pipeline. This stage allows the programmer to transform the original geometry and replace it with alternatives. When Mario encounters an enemy he has the option to throw a bomb at the enemy. The vertices of the Goomba will explode after colliding with the bomb.


Example of geometry shader

- The fragment shader is a mandatory stage in the pipeline. This shader runs once per output fragment and either sets a colour or does not draw the fragment. The fragment shader is what would give colours to our level and the character itself.

2. **Explain how the Phong lighting model allows you to create a metallic feel for objects within the game.**



- Phong lighting uses 3 components: ambient, diffuse, and specular lighting
- Even when it is dark most objects are usually not completely unlit due to a source of light somewhere in the world like a moon during the night. Ambient lighting simulates this giving objects a little bit of colour always
- Diffuse lighting simulates the lighting of the object from the source of light
- Properly done diffuse lighting will simulate the lighting and the shadows of the object based on the incoming light rays

- Phong lighting would rely on the use of specular within the lighting to create the metallic feel, as it relies on the reflectiveness of the surface to output a reflection
- Without the specular, the lighting would just be matte, which isn't very metallic at all
- A higher shininess value will contribute to a feel closer to metallic, as it is more of a spot being reflected

3. Explain what approach allows you to create a winter feel using Shaders.

- We can create particles using shaders.
- Create a tiny 2D quad that always faces the camera with a snowflake texture
- We can render all the particles using the particle emitter object to generate a single VBO with the particles and use instancing
- This way we can creating a series of falling snowflakes to simulate a winter feel
- The fragment shader will help the scene come to life with colours that represent winter such as blue, grey, and white. The opaqueness can also be adjusted in the shader to make the fragments transparent. This would allow us to add blue ice into the scene with a semi-transparent look.
- We can also use the LUT or colour ramp to modify the colour of the original texture. We can make the original texture turn more grey and blue to represent the cold weather in winter.

Explanation of how to Implement

1. **A dynamic light that gives the effect of changing the scene (e.g., day passing, seasonal changes, etc.). This includes proper light behaviour when moving away or closer to objects.**

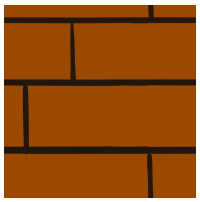
The dynamic light we would use would change colours from a very light yellow to a nice blue to represent the sun and the moon. It would also move around our scene slowly to indicate the same change. Firstly, we would set up average phong lighting. We would have to set up a loop that would check to see how much time in seconds has passed and an if statement that will check the position of the light. This same loop and if statements could also change the colour depending on which position the light is currently at.

2. **Explain how you implemented the shader for this Midterm and indicate why this choice was made.**
    a. Step one is creating a framework for the game and load in shaders, textures, and initialise the gamestate
    b. We can then render sprites into the game world using 2D projection matrix
    c. We can transform these sprites using vertex shaders
        i. For example, character can flash red when taking damage
    d. After positioning the sprites in the correct location we can draw them/render them on the screen
    e. The player character will need to have the ability to move around so we need to define those

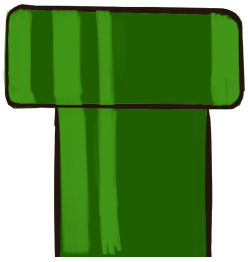
i. And we can use transformations to move the character around
ii. We just need to log the keypresses and give a value to the character velocity
iii. The jumping mechanic can be solved with a gravitational equation subtracting from the vertical velocity

Implementation

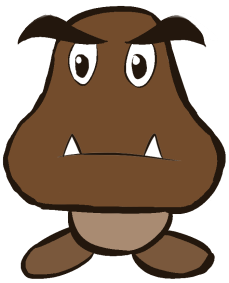
Textures



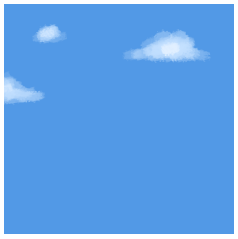
- Ground texture that is displayed throughout the whole level. It is tileable so the level can be expanded further.



- The Pipe is an obstacle for Mario to jump over and then encounter an enemy as shown in our storyboard below.



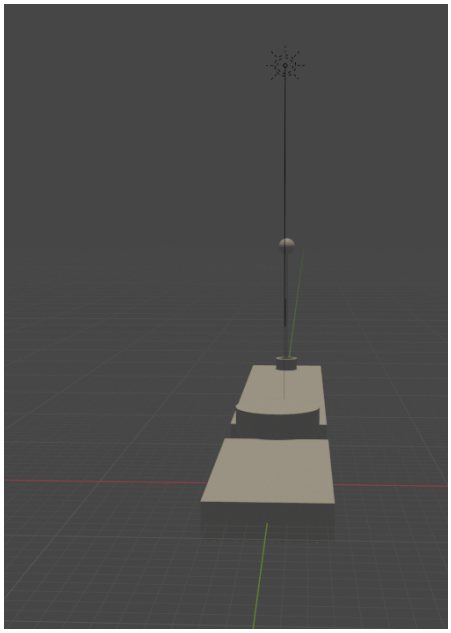
- The Goomba is a moving enemy that runs toward Mario



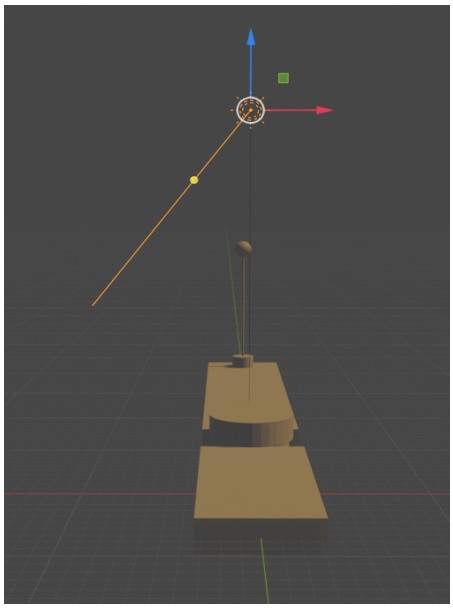
- Skybox texture is rendered in the background as the background

Basic Lighting

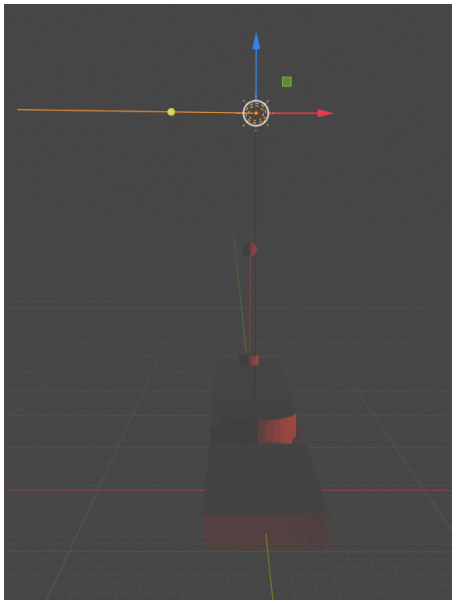
- The dynamic lighting we chose to do in our level represents the passing of time as Mario goes through the level. The shadows in the images below show the passage of time as well.
- Colours and the direction of the light changing from yellow to blue



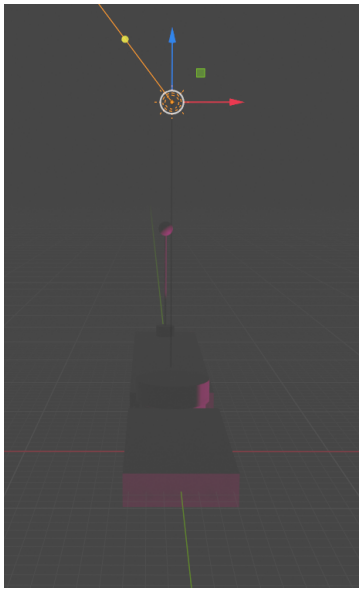
Lighting position 1: The light is yellow and is straight down



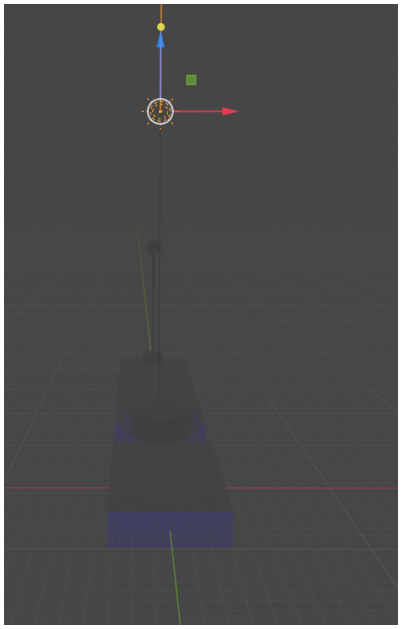
Lighting position 2: The light is orange and is Angled



Lighting position 3: The light is red and is horizontal



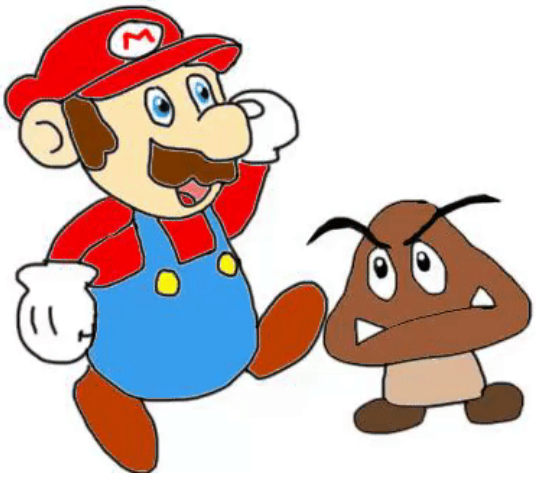
Lighting position 4: The light is purple and is Angled



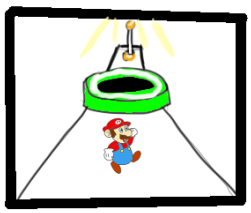
Lighting position 5: The light is blue and is Straight up

Shader

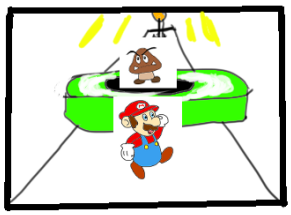
- When Mario gets hit by an enemy such as the Goomba he flashes red to show that he took damage. We would be using a fragment shader here to change the colour values to give everything a red hue. We chose this shader because the flash of a bright colour shows the player that making contact with the enemy is dangerous to Mario's health. It also helps identify who the enemy is to the player.



StoryBoard



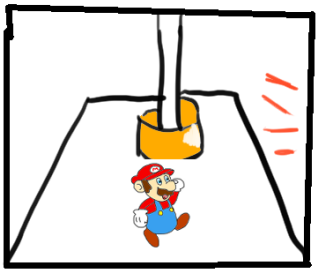
The position of Mario when starting the game



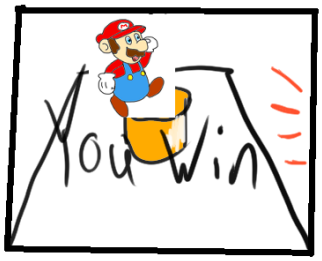
Mario as he approaches the pipe and the Goomba appears

Mario if he touches the Goomba Once

Mario if he touches the Goomba twice



Mario as he reaches the flag pole



Mario as he touches the flag pole

References

Phong Lighting Model. (n.d.). [Photograph]. Learn OpenGL.

https://learnopengl.com/Lighting/Basic-Lighting

Geometry Shader. (2020, Jan 18). [Photograph]. Unity Vertex Shader and Geometry Shader Tutorial.

https://gamedevbill.com/unity-vertex-shader-and-geometry-shader-tutorial/