Course: CSE 4208

Course Title: Computer Graphics Laboratory

Project Name: 3D BIGBEN

Submitted by,

Shrabanti Debnath Urmi

Roll: **1907094**

Year: 4th

Semester: 2nd

Group: B2

Date of Submission: 26.01.2025

Objectives:

i. To make a 3D representation of the BigBen using OpenGL.

ii. To enable movable camera controls so that the user can change the camera viewpoint.

iii. To integrate the use of advanced functionalities in OpenGL such as textures, and realistic lighting.

iv. To implement dynamic elements, such as moving birds, to enhance the visual appeal.

v. To provide customizable camera controls for exploring the scene from various perspectives.

vi. To demonstrate efficient use of OpenGL features such as lighting, texture mapping, and animations.

vii. To optimize performance for smooth rendering and user interaction.

Introduction:

The "Big Ben 3D Visualization" is a project developed using OpenGL and C++ to recreate an interactive and visually captivating model of Big Ben. The project features a detailed 3D rendering of the iconic clock tower, complemented by animated birds,water that bring movement and realism to the scene. It utilizes modern graphics libraries such as GLFW and GLAD for efficient rendering, and glm for advanced matrix transformations, providing an immersive and interactive experience.

This report outlines the objectives, features, and technical implementations of the Big Ben 3D project, emphasizing its intricate visual details and user interaction capabilities.

Characteristics:

1. **3D Model of Big Ben:**
   Described 3D architecture focusing on the famous clock and the intricate tower.
   Severe textures and architectonic material to underscore the grandeur of Big Ben.

2. **Fractal Trees and Environment:**
   Created on a fractal basis for cooling environments.

3. **Sky Dome:**
   Very real sky dome that depicts time of day, many outdoor settings.

4. **Water in motion:**
   Animation of flow from the purest dyed wave movements.

5. **Animated Birds:**
   Birds gliding about around the Big Ben, making their lovely motion.
   The speed, direction, and the type of bird motion can be adjusted; hence, dynamic nature.

6. **Dynamic Lighting:**
   Directional, point, and ambient light for an aesthetically pleasing look of a concept from many angles to give the scene aesthetic, these lights cast shadows all around.

7. **Camera Controls:**
   Visit Free Flow Camera Mode Allows full speculative inspection of Big Ben and its immediate vicinity also includes preset views for the front or back of the clock tower and the overall environment.

Output Pictures:

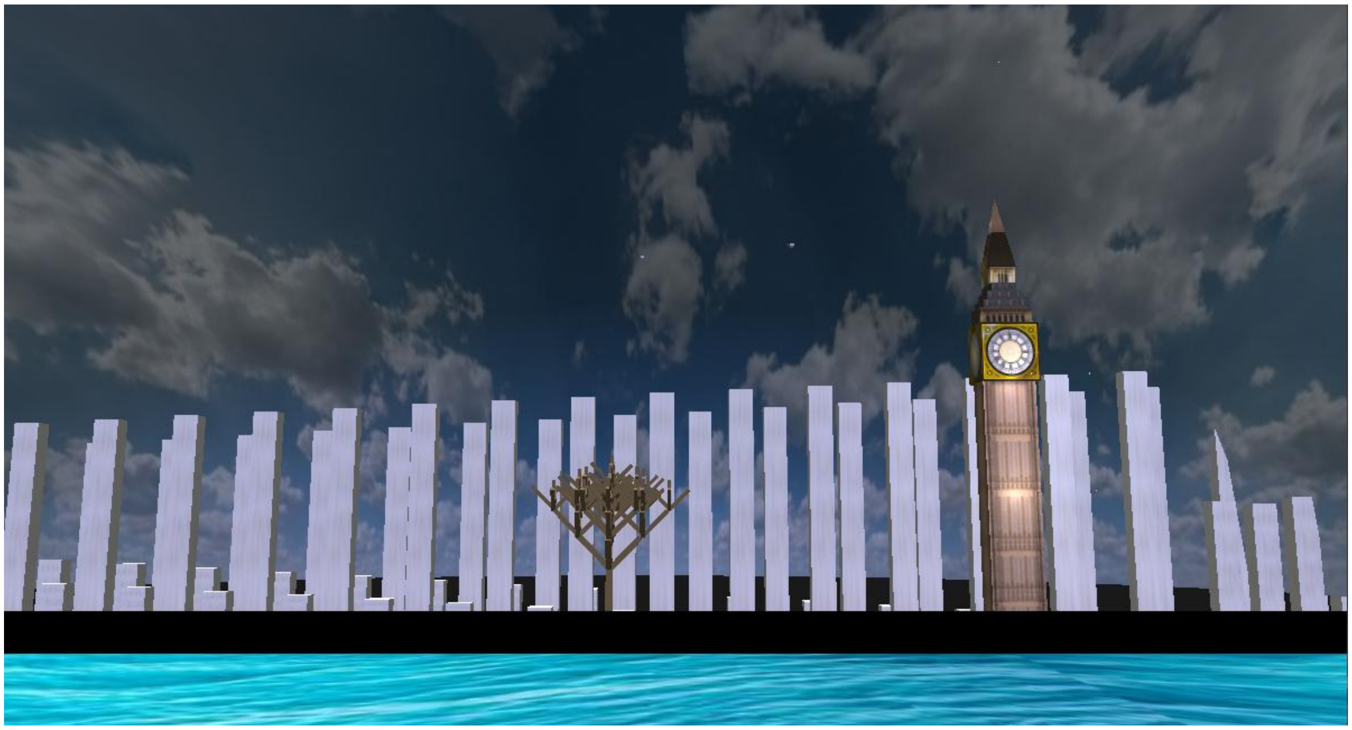

Fig 1 : Starting Scene

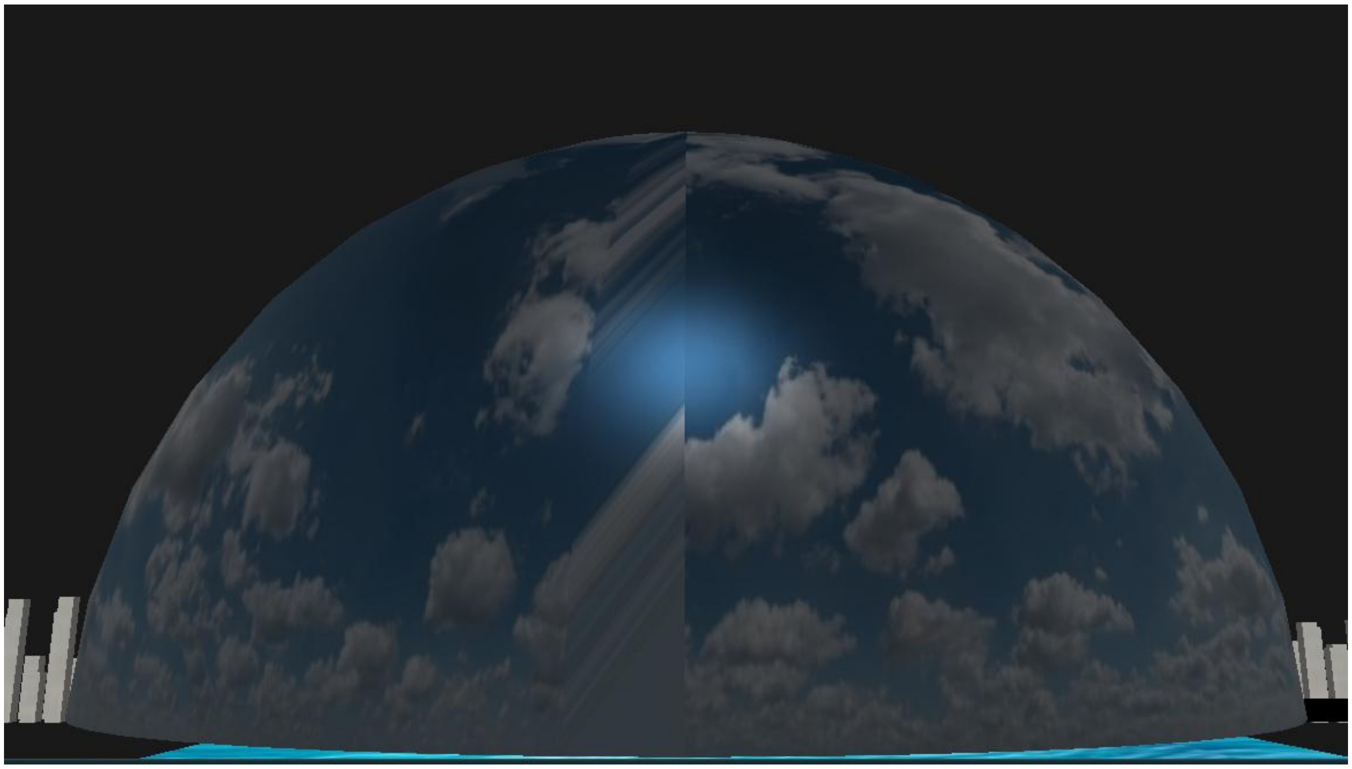

Fig 2 : Sky Dome

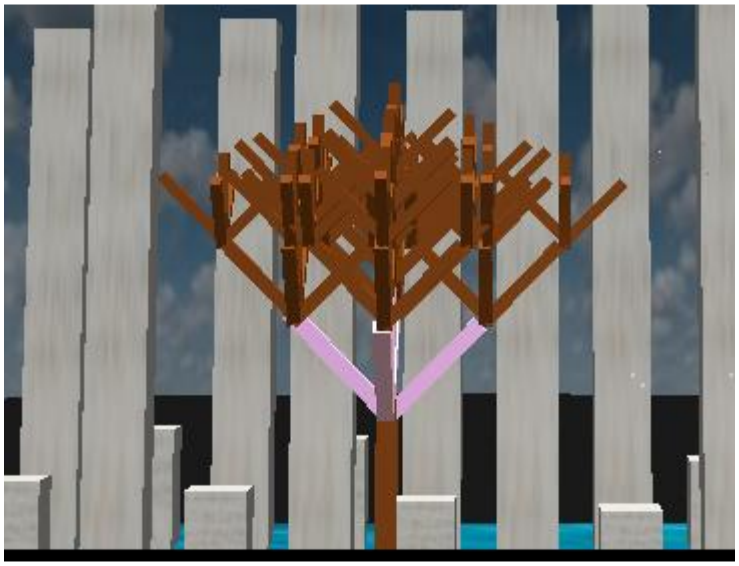

Fig 3 : Fractal Tree

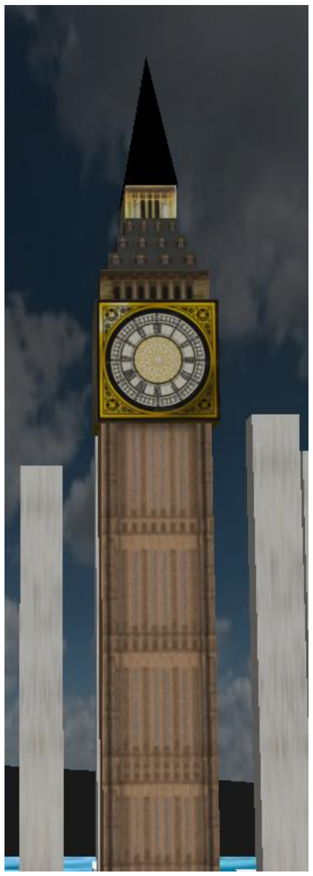

Fig 4 : BigBen

Fig 5 : Night View

Fig 6 : Moon Light View

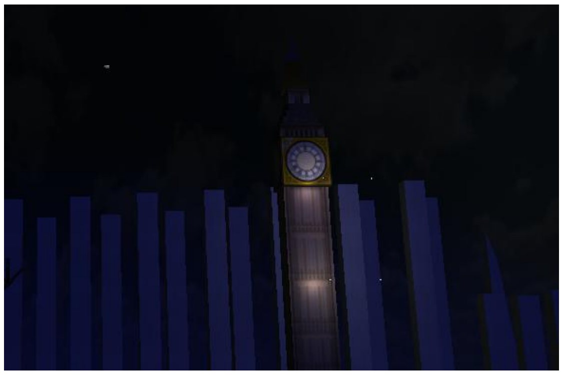

Fig 7: Point Light

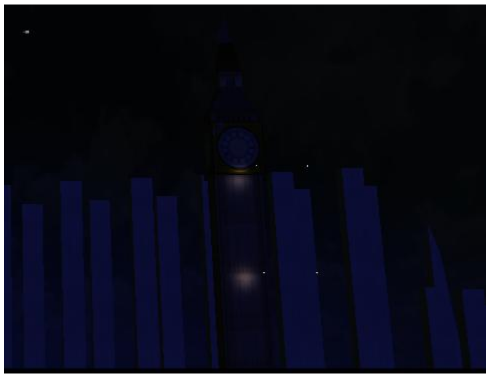

Fig 8 : Specular Light

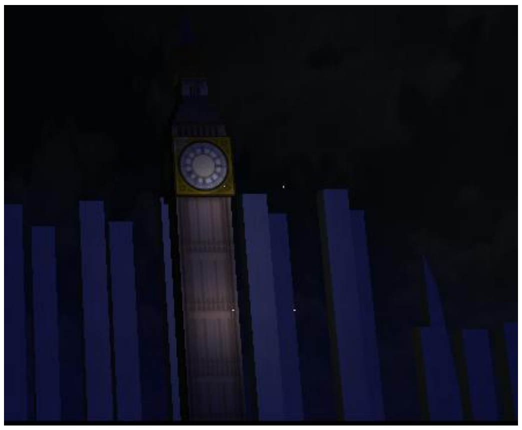

Fig 9 : Diffuse Light

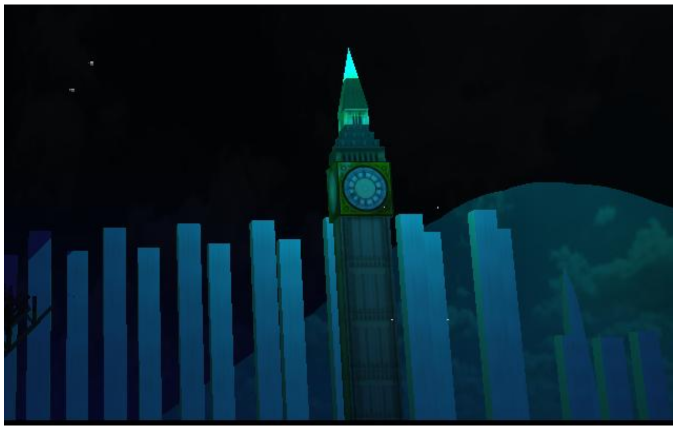

Fig 10 : Spot Light

Fig 11 : Moving Birds

Fig 12 : Moving Water

Keyboard Functionalities:

The project offers intuitive keyboard controls for navigation and interaction:

1. **Camera Controls:**
    - W / S: Move forward or backward.
    - A / D: Move left or right.
    - I / K: Pitch (up or down).
    - L / J: Yaw (right or left).

2. **Lighting Toggles:**
    - N: Day and night toggle.
    - M: Moon light toggle.
    - 1: Toggle point lights (ON/OFF).
    - 2: Toggle specular lights.
    - 3: Toggle diffuse light.
    - 4: Toggle ambient light.
    - 5: Toggle point light.

3. **Scene Controls:**
    - Esc: Exit the application.

Bird Animation Algorithm (Pseudocode):

Below is the pseudo-code for bird animation:

```
Function drawBirds(cubeVAO, lightingShader, parentModel, time):
Initialize model matrix for the bird as identity matrix
Initialize translation matrix as identity matrix
Initialize scale matrix as identity matrix
Set birdSpeed to 1.0
Set birdAltitude to 60.0
Calculate bird's x position (birdPathX) based on sine of time multiplied by birdSpeed
Calculate bird's z position (birdPathZ) based on cosine of time multiplied by birdSpeed

// Draw first bird - Body
Set translation matrix for bird's position
Set scale matrix for bird's body dimensions
Update model matrix based on parentModel, translation, and scale
Call drawCube with cubeVAO, lightingShader, model matrix, and color
```

```
// Draw first bird - Left Wing
Set translation matrix for left wing's position
Set scale matrix for left wing's dimensions
Update model matrix based on parentModel, translation, and scale
Call drawCube with cubeVAO, lightingShader, model matrix, and color

// Draw first bird - Right Wing
Set translation matrix for right wing's position
Set scale matrix for right wing's dimensions
Update model matrix based on parentModel, translation, and scale
Call drawCube with cubeVAO, lightingShader, model matrix, and color

// Second bird - Body
Calculate second bird's x position (birdPathX) based on sine of time + 3 seconds, multiplied by birdSpeed
Calculate second bird's z position (birdPathZ) based on cosine of time + 3 seconds, multiplied by birdSpeed
Set translation matrix for second bird's position at a higher altitude
Set scale matrix for second bird's body dimensions
Update model matrix based on parentModel, translation, and scale
Call drawCube with cubeVAO, lightingShader, model matrix, and color

// Second bird - Left Wing
Set translation matrix for second bird's left wing's position
Set scale matrix for second bird's left wing's dimensions
```

```
Update model matrix based on parentModel, translation, and scale

Call drawCube with cubeVAO, lightingShader, model matrix, and color



// Second bird - Right Wing

Set translation matrix for second bird's right wing's position

Set scale matrix for second bird's right wing's dimensions

Update model matrix based on parentModel, translation, and scale

Call drawCube with cubeVAO, lightingShader, model matrix, and color
```

Moving Water Pseudocode:

WaterFragmentShader.fs:

```
Function main():

// Define the output fragment color

Output FragColor as vec4

// Declare and initialize TexCoord (interpolated texture coordinates)

// Declare waterTexture as a sampler2D (the water texture sampler)

// Declare and initialize time (for wave animation)
// Calculate wave effect by modifying the x-component of the texture coordinates

Set wave to sine of (10.0 multiplied by TexCoord.x plus time multiplied by 0.5); multiply result by 0.05

// Offset texture coordinates by adding time scaled distortion and wave distortion

Set distortedTexCoord to TexCoord plus vec2(time multiplied by 0.01, wave)

// Sample the water texture using the distorted texture coordinates

Set waterColor to texture sample from waterTexture using distortedTexCoord

// Output the final fragment color

Set FragColor to waterColor
```

WaterVertexShader.vs:

```
Function main():

// Declare and initialize input variables for vertex position (aPos) and texture coordinates (aTexCoord)

// Declare output variable for texture coordinates (TexCoord)

// Declare and initialize uniform matrices: model, view, projection

// Compute the final vertex position by multiplying projection, view, and model matrices with the vertex position

Set gl_Position to the product of projection, view, model, and vec4(aPos, 1.0)

// Pass the texture coordinates to the fragment shader

 Set TexCoord to aTexCoord
```

Discussion:

The "Big Ben 3D Visualization" combines art and technology to create a detailed and realistic representation of one of the world's most iconic landmarks. With features like dynamic lighting, animated birds, it captures the beauty and motion of Big Ben while offering an interactive experience. The modular design allows for adding new features, such as real-time chimes or interactive clock adjustments.

To ensure smooth performance, especially on low-end systems, developers can use techniques like Level of Detail (LOD). Advanced shaders can improve the textures of the clock face and metallic surfaces, while realistic reflections on nearby water can enhance the overall visual quality.

Conclusion:

The "Big Ben 3D Visualization" showcases the power of OpenGL in creating immersive and interactive 3D experiences. With detailed models and dynamic animations, it highlights the architectural beauty of Big Ben. Future improvements could include adding nearby landmarks, weather effects, or VR support for a more engaging and educational experience.

References:

1. https://www.javatpoint.com/computer-graphics-tutorial
2. https://www.youtube.com/watch?v=ryBqDxJEFuU
3. https://math.hws.edu/graphicsbook/c3/index.html