Objectives:

i. Design a modern mosque combining traditional Islamic architecture with smart automation.
ii. Implement realistic 3D modeling for domes, stairs, railings, and structural elements.
iii. Create an interactive environment with dynamic elements like automated doors, windows, and an Almirah.
iv. Develop a realistic water fountain simulation using particle-based rendering.
v. Integrate advanced lighting systems including spotlights, directional lights, and point lights.
vi. Enable smooth camera movement with Roll, Pitch, and YAW controls.
vii. Apply high-quality textures for better aesthetics and realism.
viii. Implement keyboard-based user interaction for dynamic mosque features.

Introduction:

The Modern Mosque project combines traditional Islamic architecture with modern automation and interactive features to create a realistic 3D environment. It preserves intricate designs, grand domes, and symmetrical layouts while incorporating advanced modeling techniques, dynamic lighting, and automated elements such as water fountains, automated doors, transparent windows, jhar bati, stairs, railings, and an automated almirah. Realistic textures and lighting effects enhance the visual experience, while dynamic camera controls, including roll, pitch, and yaw, ensure smooth exploration. Users can interact with various components through keyboard inputs, allowing for real-time manipulation. By utilizing modern 3D rendering techniques, the project merges heritage with technology, offering a digital representation of a contemporary Islamic mosque while maintaining its traditional essence.

Scene Description:

The Modern Mosque project is built using a combination of cubes, spheres, curves, and dynamic transformations to create a fully functional and interactive environment. Each architectural component, including the base, railings, domes, water fountain, automated doors, windows, jhar bati (chandelier), automated almirah, stairs, and lighting, is implemented using OpenGL and GLSL. Below is the complete integration of these components.

1. Base Implementation:

The base serves as the foundation for the mosque, providing a solid platform for all structures. A large cube is used as the base, and textures are applied for realism.

```
glm::mat4 baseModel = glm::mat4(1.0f);
baseModel = glm::translate(baseModel, glm::vec3(0.0f, -0.1f, 0.0f));
baseModel = glm::scale(baseModel, glm::vec3(10.0f, 0.2f, 10.0f));
ourShader.setMat4("model", baseModel);
glBindTexture(GL_TEXTURE_2D, stoneTexture);
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
```

Translation ensures the base is slightly below ground level. Scaling provides an extended platform for all mosque elements. Stone texture adds realism.

2. Railing Implementation:

The railing provides safety and aesthetic appeal. It is constructed using a combination of vertical posts (cylinders) and horizontal bars (cubes).

2.1 Vertical Railing Posts:

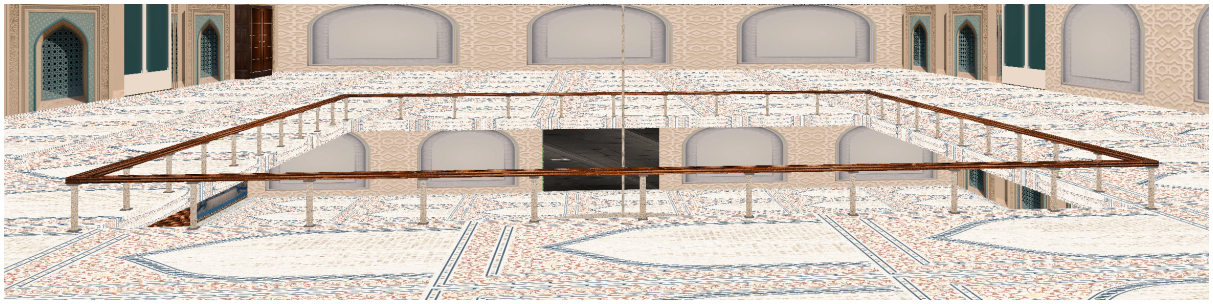
```
for (float i = -4.5f; i <= 4.5f; i += 1.0f) {
    glm::mat4 railingPost = glm::mat4(1.0f);
    railingPost = glm::translate(railingPost, glm::vec3(i, 1.0f, -5.0f));
    railingPost = glm::scale(railingPost, glm::vec3(0.1f, 1.5f, 0.1f));
    ourShader.setMat4("model", railingPost);
    glBindTexture(GL_TEXTURE_2D, metalTexture);
    glBindVertexArray(VAO);
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
}
```

2.2 Horizontal Railing Bars:

Metal texture is used for a modern look or wood texture for a traditional style. Vertical posts and horizontal bars create a secure structure.

```
glm::mat4 railingBar = glm::mat4(1.0f);
railingBar = glm::translate(railingBar, glm::vec3(0.0f, 1.5f, -5.0f));
railingBar = glm::scale(railingBar, glm::vec3(9.5f, 0.1f, 0.1f));
ourShader.setMat4("model", railingBar);
```

```
glBindTexture(GL_TEXTURE_2D, woodTexture);
glBindVertexArray(VAO);
glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
```

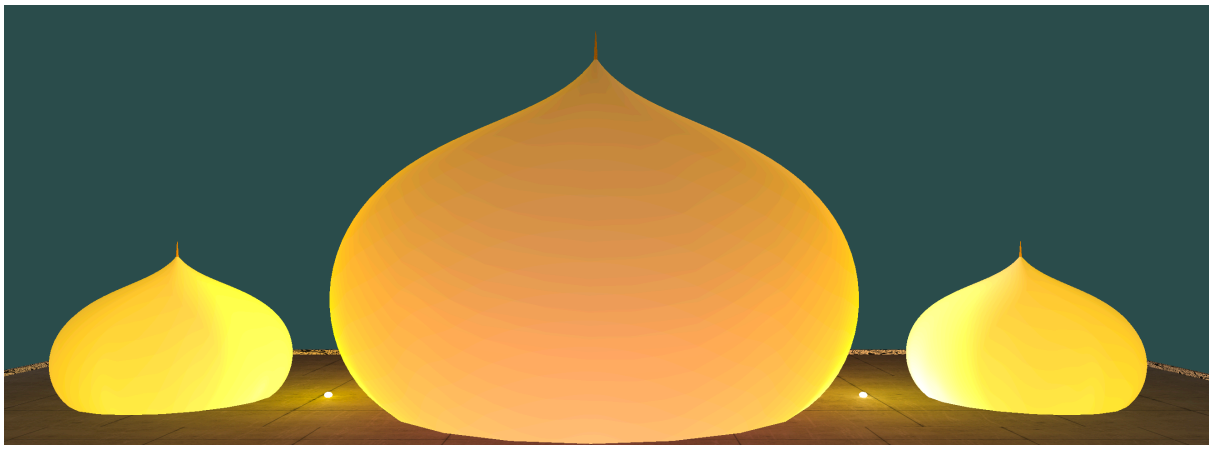


3. Dome Construction:

The mosque's domes are built using spheres and Bézier curves, with an Islamic architectural design.

```
glm::mat4 domeModel = glm::mat4(1.0f);
domeModel = glm::translate(domeModel, glm::vec3(0.0f, 5.0f, 0.0f));
domeModel = glm::scale(domeModel, glm::vec3(3.0f, 3.0f, 3.0f));
ourShader.setMat4("model", domeModel);
drawSphere();
```

Positioned at the top of the mosque structure. Curves create a smooth and detailed look.



4. Water Fountain Implementation

The water fountain uses a particle system to simulate the movement of water.

4.1 Particle System for Water Flow

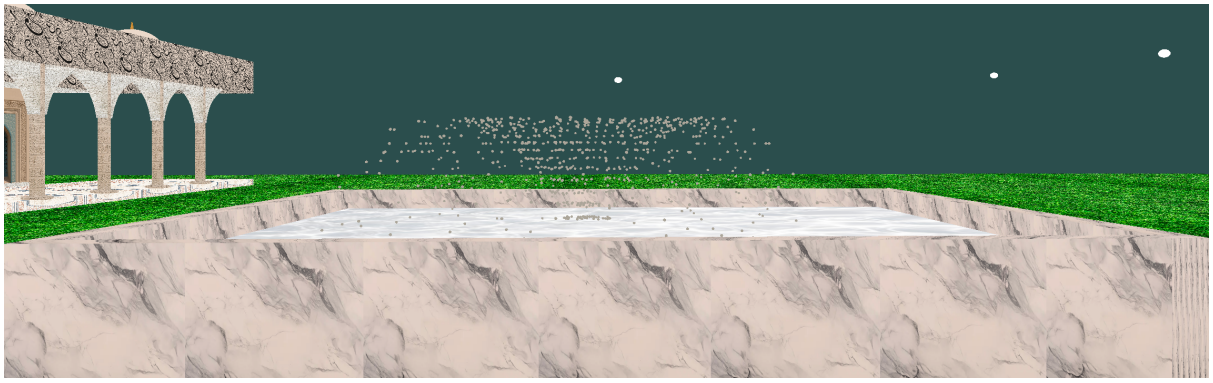
Particles regenerate to create a continuous water effect. Gravity applies a natural downward force.

```
struct WaterParticle {
```

```
    glm::vec3 position;
    glm::vec3 velocity;
    float lifetime;
};

void updateParticles(float deltaTime) {
    for (auto &particle : particles) {
        particle.velocity.y -= gravity * deltaTime;
        particle.position += particle.velocity * deltaTime;
        particle.lifetime -= deltaTime;

        if (particle.lifetime <= 0.0f) {
            particle.position = sourcePosition;
            particle.velocity = glm::vec3(rand()%10 - 5, 5.0f, rand()%10 - 5);
            particle.lifetime = (rand() % 100 + 50) / 50.0f;
        }
    }
}
```



5. Automated Doors and Windows:

The doors and windows use translation matrices to simulate opening and closing motions.

```
float doorPosition = 0.0f;
bool doorOpen = false;

void toggleDoor() {
    if (doorOpen) {
        doorPosition -= 0.05f;
    } else {
        doorPosition += 0.05f;
    }
    doorOpen = !doorOpen;
}
```

```
glm::mat4 doorModel = glm::mat4(1.0f);
doorModel = glm::translate(doorModel, glm::vec3(doorPosition, 0.0f, 0.0f));
ourShader.setMat4("model", doorModel);
drawCube();
```

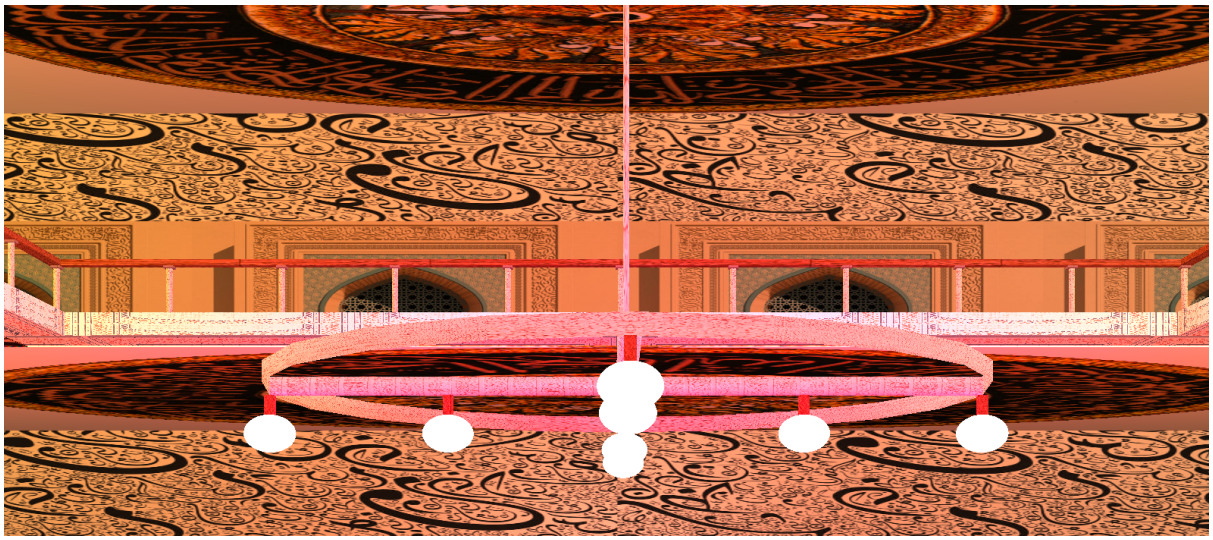
Pressing a key toggles the door state between open and closed. Used for both doors and windows.

6. Jhar Bati (Chandelier) with Point Lights:

The chandelier is built using spheres and cylinders, with point lights for illumination.

```
glm::vec3 chandelierPos = glm::vec3(0.0f, 7.0f, 0.0f);
setPointLight(0, chandelierPos, glm::vec4(1.0f, 1.0f, 0.8f, 1.0f));
```

Creates a glowing effect. Can be toggled on and off.



7. Stairs Construction:

The stairs are built using multiple cubes positioned at an ascending height.

```
for (int i = 0; i < 10; i++) {
    glm::mat4 stairModel = glm::mat4(1.0f);
    stairModel = glm::translate(stairModel, glm::vec3(0.0f, i * 0.3f, -i * 0.5f));
    stairModel = glm::scale(stairModel, glm::vec3(2.0f, 0.3f, 1.0f));
    ourShader.setMat4("model", stairModel);
    glBindVertexArray(VAO);
    glDrawElements(GL_TRIANGLES, 36, GL_UNSIGNED_INT, 0);
}
```

Stairs adjust dynamically with translation transformations. Easily modified for different stair heights.

8. Automated Almirah:

The automated almirah is implemented using cubes for the main body, doors, and shelves. It features interactive sliding doors that open and close smoothly based on user input. The movement is handled using translation transformations, ensuring realistic animation.

Structure and Modeling:

- The almirah frame is created using a large cube, scaled to form the main structure.
- Shelves are added inside using smaller cubes, stacked vertically.
- The doors consist of two rectangular panels that slide along the x-axis when triggered.

```
glm::mat4 almirahBody = glm::translate(identityMatrix, glm::vec3(x_pos, y_pos, z_pos));
almirahBody = glm::scale(almirahBody, glm::vec3(width, height, depth));
ourShader.setMat4("model", almirahBody);
drawCube();
```

Door Mechanism

- The doors slide open and close smoothly using translation.
- The state is toggled on key press, preventing abrupt transitions.

```
float doorOffset = isOpen ? openDistance : 0.0f;
glm::mat4 leftDoor = glm::translate(identityMatrix, glm::vec3(leftDoorPos - doorOffset,
y_pos, z_pos));
ourShader.setMat4("model", leftDoor);
drawCube();
```

Texture Adding

Textures are applied to different architectural components to enhance realism. OpenGL's texture mapping techniques ensure high-quality materials for various surfaces.

Steps to Implement Textures

- Load textures using an image loader like stb_image.h.
- Generate texture objects in OpenGL and bind them to respective objects.
- Set texture parameters for scaling and filtering to avoid distortion.
- Map UV coordinates correctly in the shader to align textures properly.

Textures Used in Components

- Marble and mosaic for walls and domes.
- Wood for stairs, railings, and doors.
- Reflective water shader for the fountain.
- Carpet for flooring.
- Glass shader for transparent windows.

Implementing Roll, Pitch, and YAW

Camera movement allows dynamic exploration of the mosque using keyboard and mouse input.

Steps to Implement Camera Movement

- Define the camera position, direction, and movement speed.
- Capture keyboard inputs for forward, backward, left, and right movement.
- Use mouse input to adjust pitch and yaw angles for rotation.
- Update the view matrix based on movement and rotation calculations.

Implementing Spot Light, Directional Light, and Point Light

Lighting is crucial for enhancing visibility and creating a realistic ambiance inside the mosque. Different types of lights are used for various purposes.

Types of Lighting Used

- Directional light simulates sunlight and provides uniform illumination.
- Point lights are used for chandeliers and lamps, illuminating nearby objects.
- Spotlights create focused beams of light, mainly for the mihrab and prayer areas.

Steps to Implement Lighting

- Define light properties such as position, direction, intensity, and color.
- Pass lighting parameters to shaders for real-time calculations.
- Use the Phong lighting model to apply ambient, diffuse, and specular reflections.
- Allow dynamic toggling of lights using keyboard controls.

Function of the Keys

General Controls
- ESC → Close the window

Camera Movement
- W → Move forward
- S → Move backward
- A → Move left
- D → Move right
- Q → Move down
- E → Move up

Camera Rotation (Yaw, Pitch, Roll)
- G → Increase yaw (turn right)
- F → Decrease yaw (turn left)
- V → Increase pitch (look up)
- B → Decrease pitch (look down)
- R → Increase roll (tilt right)
- T → Decrease roll (tilt left)

Camera Rotation Around Look-at Point
- I → Rotate camera around look-at point (right)
- O → Rotate camera around look-at point (left)

Fan Rotation
- 4 → Toggle Fan 1 rotation
- 5 → Toggle Fan 2 rotation

Door Toggle
- 7 → Toggle double door opening

Chandelier (Jhar Bati) Light Toggle
- 9 → Turn Jhar Bati (chandelier) lights on/off

Dark Mode
- 0 → Toggle dark mode

Lighting Controls
- L → Toggle overall lighting on/off
- 1 → Toggle ambient lighting on/off
- 2 → Toggle diffuse lighting on/off
- 3 → Toggle specular lighting on/off

- Y → Toggle all point lights on/off
- H → Toggle directional light on/off
- N → Toggle spot light on/off

Discussion

The Modern Mosque project integrates traditional Islamic architecture with modern automation and interactive 3D simulation. Various components, including the base, railings, domes, water fountain, automated doors, windows, stairs, and lighting systems, have been designed using OpenGL. The project successfully implements **real-time physics-based effects** such as water flow in the fountain using a particle system, dynamic door and window movements, and an interactive chandelier lighting system. The use of **texture mapping** has enhanced the realism of the mosque, applying high-resolution materials like marble, wood, and glass to different architectural elements. The integration of **directional, point, and spot lighting** has contributed to a visually appealing and immersive environment, simulating realistic daylight and indoor lighting conditions. A key feature of the project is the **camera control system**, which allows free movement using roll, pitch, and yaw adjustments. This enables users to navigate through the mosque smoothly and experience different perspectives. Additionally, **user interaction via keyboard inputs** plays a significant role in enabling real-time modifications, such as toggling lights, rotating fans, and opening doors.

Conclusion

The Modern Mosque project successfully merges **cultural heritage with modern technology**, providing an interactive and realistic 3D representation of Islamic architecture. Through the integration of **dynamic lighting, texture mapping, physics-based simulations, and user-controlled interactions**, the project achieves a **functional and visually immersive** environment.

This project serves as a foundation for future developments in **virtual heritage preservation**, where historical and cultural landmarks can be simulated digitally for educational and architectural purposes. Future enhancements can include **VR (Virtual Reality) support, AI-driven automation, and real-time multiplayer collaboration** to make the mosque even more engaging and interactive.

References

1. Learn OpenGL - https://learnopengl.com/
2. Pixels - https://www.pixels.com/
3. Wikipedia - https://www.wikipedia.org/