Universidad Nacional de Ingeniería (UNI)

Recinto Universitario Simón Bolívar (RUSB)

Área del conocimiento de Tecnologías de información y Comunicaciones (TIC)

Target Practice

Collaborators: Velásquez Chavarría Rommel Jovanny.

Herrera Dominguez Cristopher Aaron.

Norori Amador Engell Mckenzic.

ID Number: 2022-0316U

2020-0488U

2022-0279U

Class: 3T3-CO

Instructor: Chavez Miranda Danny Oswaldo

Thursday, July 4, 2024

Introduction:

The project "Target Practice" was developed as a prototype to implement and demonstrate various advanced computer graphics concepts using OpenGL and C++. This project integrates multiple libraries, including stb_image, GLFW, glm, SFML, glad, and json. The main objective was to create an interactive application allowing the user to shoot in a 3D environment, implementing sound, 3D models, textures, lighting, keyboard and mouse interaction, and a skybox. The development was done in Visual Studio Community 2019.

Objectives:

- **Sound:** Implement sound effects for shooting and other events.
- **3D Models:** Load and render 3D models.
- **Textures:** Apply textures to 3D models.
- **Lighting:** Implement lighting techniques to enhance realism.
- **Keyboard and Mouse Interaction:** Allow user interaction with the application using keyboard and mouse.
- **Skybox:** Create a skybox to enhance the visual environment.

Project development:

The project development included setting up the initial development environment using Visual Studio Community 2019 and the mentioned libraries. A main loop was created using GLFW for window management, and OpenGL was configured to use the CORE profile and version 3.3. The necessary shaders (default.vert, default.frag, skybox.vert, skybox.frag, objectInstancing.vert) were loaded for rendering objects and the skybox.

```
// Generates Shader objects
Shader shaderProgram("default.vert", "default.frag");
Shader skyboxShader("skybox.vert", "skybox.frag");
Shader multipleShader("objectInstancing.vert", "default.frag");
```

Objects of models (Model) were created to represent elements such as the terrain (earthCube), trees (tree), targets (target), a character (steve), and a first-person weapon (pov). Each model is loaded from GLTF files located in the project's resource directory.

```
// Load in models
Model earthCube((parentDir + earth_object).c_str(), earthCubeNumber, earthCubeInstanceMatrix);
Model tree((parentDir + tree_object).c_str(), treeNumber, treeInstanceMatrix);
Model target((parentDir + target_object).c_str(), targetNumber, targetInstanceMatrix);
Model steve((parentDir + steve_object).c_str(), steveNumber, steveInstanceMatrix);
Model pov((parentDir + weapon_pov_object).c_str());
```

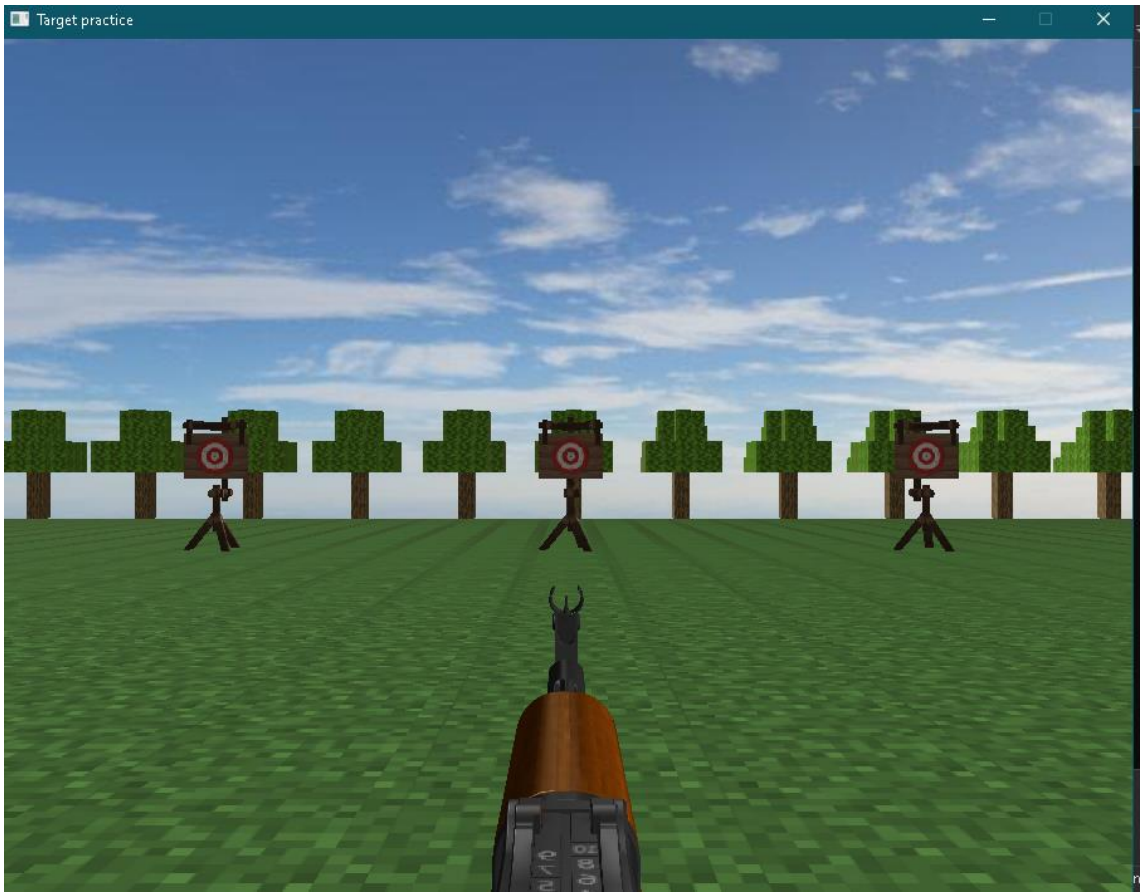
Transformation matrices were implemented to position, rotate, and scale each instance of the models in the 3D world.

```
//Positions for the earth cube
std::vector<glm::vec3> earthCubePositions;

//Fill the positions
for (float i = -52.0f; i <= 52.0f; i += 2.0f) {
    for (float j = -52.0f; j <= 52.0f; j += 2.0f) {
        //Add positions on the edge
        earthCubePositions.push_back(glm::vec3(j, 0.0f, i));
    }
}

const unsigned int earthCubeNumber = earthCubePositions.size();
std::vector <glm::mat4> earthCubeInstanceMatrix;
for (size_t i = 0; i < earthCubeNumber; i += 1) {
    glm::quat rotation = glm::rotate(glm::mat4(1.0f), glm::radians(90.0f), glm::vec3(0.0f, 0.0f, 1.0f));
    glm::mat4 rot = glm::mat4_cast(rotation);
    glm::mat4 sca = glm::scale(glm::mat4(1.0f), glm::vec3(1.0f, 1.0f, 1.0f));
    glm::mat4 trans = glm::translate(glm::mat4(1.0f), earthCubePositions[i]);

    earthCubeInstanceMatrix.push_back(trans * rot * sca);
}
```

A camera (Camera) was used to control the player's view and orientation, integrating keyboard and mouse functions directly into the camera class to manage user interaction.

The camera allows the user to navigate the environment using the following inputs:

- **A**: Move left
- **W**: Move forward
- **S**: Move backward
- **D**: Move left
- **K**: Shoot



Execution Preview.

Textures and materials were configured for each model, and basic lighting techniques were implemented using a lighting shader (`default.vert`, `default.frag`) and a specific shader for the skybox (`skybox.vert`, `skybox.frag`).

```
// Take care of all the light related things
glm::vec4 lightColor = glm::vec4(1.0f, 1.0f, 1.0f, 1.0f);
glm::vec3 lightPos = glm::vec3(0.5f, 0.5f, 0.5f);
shaderProgram.Activate();
glUniform4f(glGetUniformLocation(shaderProgram.ID, "lightColor"), lightColor.x, lightColor.y, lightColor.z, lightColor.w);
glUniform3f(glGetUniformLocation(shaderProgram.ID, "lightPos"), lightPos.x, lightPos.y, lightPos.z);
skyboxShader.Activate();
glUniform1i(glGetUniformLocation(skyboxShader.ID, "skybox"), 0);
multipleShader.Activate();
glUniform4f(glGetUniformLocation(multipleShader.ID, "lightColor"), lightColor.x, lightColor.y, lightColor.z, lightColor.w);
glUniform3f(glGetUniformLocation(multipleShader.ID, "lightPos"), lightPos.x, lightPos.y, lightPos.z);
```

Rendering was optimized using techniques such as face culling (glCullFace) and enabling the depth buffer (glEnable(GL_DEPTH_TEST)) to improve performance and visual quality.

```
// Enables the Depth Buffer
glEnable(GL_DEPTH_TEST);

// Enables Cull Facing
glEnable(GL_CULL_FACE);
// Keeps front faces
glCullFace(GL_FRONT);
// Uses counter clock-wise standard
glFrontFace(GL_CCW);
```

Conclusion:

In conclusion, the "Target Practice" project successfully demonstrated the integration of 3D graphics technologies and C++ programming. It achieved the implementation of an interactive environment where users can explore, aim, and shoot in a realistic virtual environment. Various key aspects of OpenGL were explored, ranging from initial setup to advanced implementation of visual effects and rendering techniques.

Bibliography.

Gordan, V. (2021, april). OpenGL Tutorial 13 - Model Loading [Video]. YouTube. https://youtu.be/AB_f4slL2H0

Gordan, V. (2021, jun). OpenGL Tutorial 19 - Cubemaps & Skyboxes [Video]. YouTube. https://youtu.be/8sVvxeKI9Pk?si=pgA8lOf8kOjBdZEw

Gordan, V. (2021, july). OpenGL Tutorial 21 - Instancing [Video]. YouTube. https://youtu.be/TOPvFvL_GRY?si=_erA--F7_AyNiBX7

Appendices.

Architecture diagram.