|  |  |
| --- | --- |
|  | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

Computer Graphics Project Report

**Solar System**

**Project Group Members:**

Mohammed Al-Eida (201807140)

Mohammed Tahar (201610551)

Adel Hasan (201905244)

**Instructor**: Dr.Osama Halabi

**2023**

Table of Contents

[List of Figures iii](#_Toc152877928)

[List of Tables iv](#_Toc152877929)

[1. Project Goals 5](#_Toc152877930)

[2. Project Accomplishments 5](#_Toc152877931)

[2.1. Implemented Features 5](#_Toc152877932)

[2.2. Relation to Computer Graphics 5](#_Toc152877933)

[3. Technical Details 6](#_Toc152877934)

[3.1. Rendering a Sky map around the solar system 6](#_Toc152877935)

[3.2. Interactivity 6](#_Toc152877936)

[4. Additional Instructions for Running the Code 7](#_Toc152877937)

[5. Resources Used 8](#_Toc152877938)

[6. Challenges Faced 8](#_Toc152877939)

[7. References 9](#_Toc152877940)

# List of Figures

[Figure 1: Platform Toolset setup 7](#_Toc152872427)

# List of Tables

[Table 1: Controls 7](#_Toc152872422)

# Project Goals

The primary goal of our project is to create an interactive and visually appealing 3D representation of the solar system. We aim to incorporate at least two key areas of computer graphics, such as rendering, interaction and object transformation to enhance the user engagement.

# Project Accomplishments

## Implemented Features

* Solar System Model: We imported 3D models of the solar system, including the sun and planets, and attached a texture to each object we imported.
* Rendering Techniques: To enhance realism, we implemented shaders and texture mapping. We created a sky sphere that has a space texture attached for simulating space environment. Furthermore, to simulate a realistic solar system we implemented directional light from the sun to light the planets and the sides of the planet that are facing away from the sun would be dark using light shaders.
* Transformation: We added translating motion to our planets models that enabled them to orbit around the sun in circular motion. To make the solar system look more realistic we applied rotation motion to the planets depending on their axis tilt.
* Interactivity: Users can interact with the solar system. For example, we created a small interface GUI that displays instructions to interact with the Solar system simulation. For instance, we mapped the keys from 1 to 8 to each planet respectively, when pressing any key from 1 to 8 takes the FPS camera to a close-up position to the planet that is mapped to that key. Moreover, we mapped 9 and 0 to have a view of the sun and solar system respectively. Also, when pressing the comma (,) key, it will decrease camera movement speed and when pressing period (.) key, it will increase the movement speed of the camera. Additionally, when pressing the P key, it will pause the planets from orbiting around the sun and display them in a line. Finally, users can use the WASD keys to move in around the solar system and use the mouse to control the camera while using Z and X to move up and down.

## Relation to Computer Graphics

* Our project heavily focuses on rendering to create a realistic solar system. The use of various shaders like vertex and fragment shaders and texture mapping contributes to the visual fidelity of the solar system model.
* Interactivity is a key aspect of our project. Users can manipulate the camera view through selecting the planets using the mapped hotkeys, providing an immersive experience.
* Transformations are used to provide near accurate and realistic representations of the planets while applying an axis tilt rotation so that the planets rotate always at an angle like they do in space.

# Technical Details

## Rendering a Sky map around the solar system

To simulate the vastness of space, we implemented rendering a sky sphere. This technique involves projecting the scene onto a huge sphere and having the solar system inside the sphere, providing a convincing backdrop for the solar system.

## Interactivity

To improve the user experience, we have decided to implement a GUI Interface using the “Imgui” library to display the Instructions to use the simulation as a small window. We mapped certain keys to do certain things such as pausing/unpausing the planets or having close ups of the planets. Below are the Instructions to the simulation of the Solar System:

* **Close-up View:** **keyboard keys** [**1** ,**2** ,**3** ,**4** ,**5** ,**6** ,**7** ,**8** ,**9**] are mapped to give a close-up of the planets. By clicking them we stop the Orbital rotation and only allow Self-rotation of the planets, we stop the “FREE camera Movement” using “WASD”, and bring the camera closer to the object so that the User gets a nice close view.
* **Solar System View:** by clicking the **keyboard key** “0” , same effects as the close-up view, it just give a view from above of the solar system.
* **Free Camera mode:**  once the user clicks the key “**V**”, the “WASD” movement will beunlocked taking them back into free movement mode.
* **Scrolling up/down**: will increase/decrease the Field of view.
* “**WASD**”: is used to move around, Forward/Left/Backward/Right.
* “**Z and X**”: are used to move vertically Up and down.
* “**P**”: Once clicked, Orbital movement around the sun is stopped and planets are lined up for a clear representation of how the planets are ordered from closest to furthest from the sun.
* “**,**” and “ **.** ” : are used to increase/decrease the movement speed. This is done by adding a global variable called SPEED that is increased/decreased by 2.0 then used to multiply the MOVE\_SPEED constant in the fpsCamera movement using WASD. An “If” statement used to set SPEED value to 1.0 by default once it is decreased under 1.0.

Table : Controls

|  |  |
| --- | --- |
| Key | Function |
| WASD | move Forward/Left/Backward/Right |
| Z/X | Vertical up/down |
| 1~9 | Close up of planets (locks movement) |
| 0 | Solar System view |
| P | Pause/Continue Orbit movement, to see planets in order |
| V | Free camera mode (unlocks movement) |
| Comma “,” / Period “.” | Decrease/ Increase WASD movement speed 2x |
| Scroll up/down | Zoom in/out, FOV increase/Decrease |

# Additional Instructions for Running the Code

* **Platform Toolset setup:** Once you run import the solution, go to Project> Project properties > Configuration Properties >General > Platform Toolset, and set it to the platform toolset that you have. Tested and ran on v142 and v143. A screenshot of a computer program

  Description automatically generated

Figure : Platform Toolset setup

* Make sure visual studio Debug is running on **x86**.

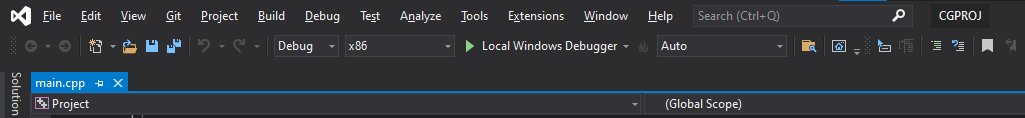
****

Figure : Platform x86

* Do not right click and include the models, same issue was noticed during the lecture of importing 3D models that when the models file content is included the project stops working.

# Resources Used

* Planet Models and Textures: Sourced from various online platforms.
* Imgui Library: Utilized for rendering a GUI interface within the glfw window.
* Sky Box Implementation Resources: Initially followed videos and guides for cube mapping but eventually opted for a sky sphere due to implementation challenges.
* Lecture Code Adaptation: Utilized the lecture code as a foundation, and we modified some parts. Such changes are the changes made to lighting shader files, importing objects as well as the texture.cpp and mesh.cpp files that we used to apply a texture to an object model.

# Challenges Faced

* Skybox Implementation: After various trials and errors while using multiple online videos and guides as references to implement a Skybox, the errors persisted. Thus, we opted to implement a Sky-Sphere instead.
* Visual Studio compatibility errors and GitHub issues: Errors appeared when sharing code due to using different Visual Studio releases. Furthermore, we encountered some challenges when we shared code using GitHub such as files not pushing properly due to Visual Studio.
* Realistic simulation and Orbital Rotation: Finding correct numbers and angles to simulate the Translation, Scaling, and Rotation of the planets as realistically as possible.

# References

1. <https://sketchfab.com/tags/planets>
2. <https://www.turbosquid.com/Search/3D-Models/free/planet>
3. <https://www.freepik.com/free-photos-vectors/space-textures>
4. <https://www.youtube.com/watch?v=QYvi1akO_Po&ab_channel=BrianWill>
5. <https://learnopengl.com/Advanced-OpenGL/Cubemaps>