Tribhuwan University

Institute of Engineering

Pulchowk Campus

A Project Report on:

# Gravity Simulation


**Submitted By:**

Amrit Baral(076bct006)

Nabin Da Shrestha(076bct037)

Nirajan Bekoju(076bct039)

Nishant Luitel (076bct041)


**Submitted To:**

Department of Electronics and Computer Engineering

**Submission Date:**

$25^{th}$ August, 2022

# Acknowledgement

Foremost we want to express our deepest gratitude to our lecturer, Dr. Basanta Joshi for his detailed guidelines for this project. His valuable lectures and lab sessions has been fundamental cornerstones for this project.

We would like to recognize the enormous influence of past projects from our seniors that gave us the motivation to come up with this project. Also, our appreciation for all the fellow friends who have offered us to help in time of need.

We also wish to thank the entire Department of Electronics and Computer Engineering, Pulchowk campus for giving students early experience in implementing their own ideas by working on a real project.

# Contents

# Abstract

Gravity simulation is the mathematical modelling of gravity. With the use of OpenGL C++, 3D rendering of all heavenly bodies as well as artificial satellites and spaceship to simulate gravity is created in this project. Normal Mapping for texture was applied in order to render the surface more clearly. Moreover, Implementation of ambient, diffuse and specular light can be observed in the project. The main goal of this project was to explore the graphics algorithms, learn OpenGL, use lighting model effectively and 3D modelling of objects. Gravity simulation: mathematical model is a complex model with lots of parameters as the motion of planets are affected by lots of parameters like rotation time of planet, distance of planet from nearby heavenly bodies, composition and densition of planets nearby and itself. In this project, stars, planets and moons can be added manually and see the effect of gravity in those heavenly bodies.



**Keywords:** Gravity, Simulation, Matrix Stack, Sky Box, Normal Map, Lighting Model


# Introduction

Gravity simulation, a computer graphics project is designed to model the working of gravity between stars, planets, moons and satellites. This project was primarily focussed on the implementation of 3D rendering of graphics and various graphics algorithms such as Z-buffer, sphere development, cube mapping and to know more about graphics language, OpenGL C++. Gravity simulation is a difficult mathematical model with lots of variables in it. For example: rotation and revolution speed of any planets or moon depends on distance of planet from nearby heavenly bodies, rotation time of the planet, composition and density of planet itself, axial tilt of a planet, orbital eccentricity of planet, etc.

# Objectives

1. To know how 3D graphics rendering works.
2. To simulate the motion of stars, planets, satellites and moons.
3. To learn graphics language, OpenGL C++.
4. To learn graphics algorithm like Z-buffer, Cube Mapping, Texture Mapping, etc.

# Methodology

## Techonologies Used

OpenGL C++ (Version 4.3) was used to develop this project. GNU Image Manipulation Program (GIMP) was used to develop normal mapping of all the texture image.

## Algorithm Used

1. Parameter Calculation

   $\vec{F} = (\frac{GMm}{R^3})\vec{r}$

   $\vec{a} = \frac{\vec{F}}{m} = (\frac{GM}{R^3})\vec{r}$

   For n bodies in system,

   $\vec{a_i} = \vec{a_{0i}} + G\sum_{i=1}^{n} \frac{M_i}{R_i^3}\vec{r_i}$

   From $\vec{a_i}$ calculate the velocity as

   $\vec{v_i} = \vec{v_{0i}} + c_i\vec{a_i}\Delta t$

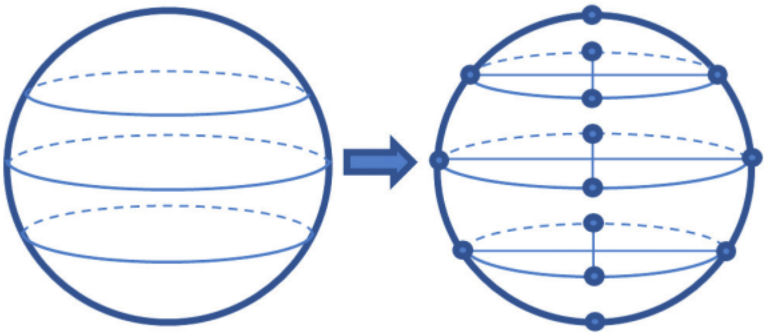
   From $\vec{v_i}$ calculate the position as

   $\vec{P_i} = \vec{P_{0i}} + b_i\vec{v_i}\Delta t$

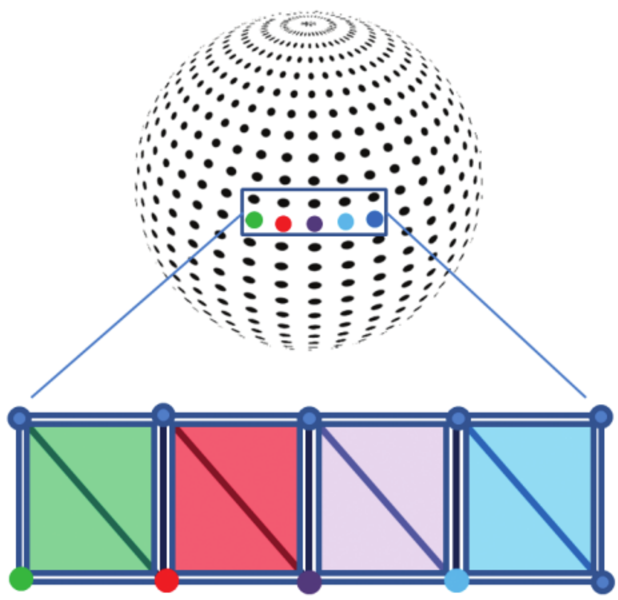
   Here $c_i$ and $b_i$ can be adjusted for different heavenly bodies.

2. Sphere Drawing Algoritm

   Step 1: Subdivide the circumference of each circular slice into some number of points. More points and horizontal slices produces a more accurate and smoother model of the sphere.

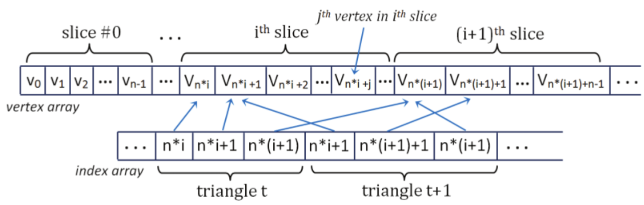


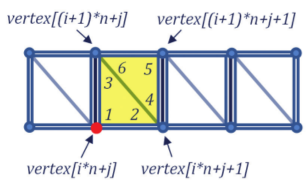
Step 2: Step through the vertices, building two triangles at each step.



Step 3: For example, move along the row of the five colored vertices on the sphere in figure shown for each of those five vertices we build the two triangles shown in the corresponding color.

Step 4:




for each horizontal slice in the sphere {

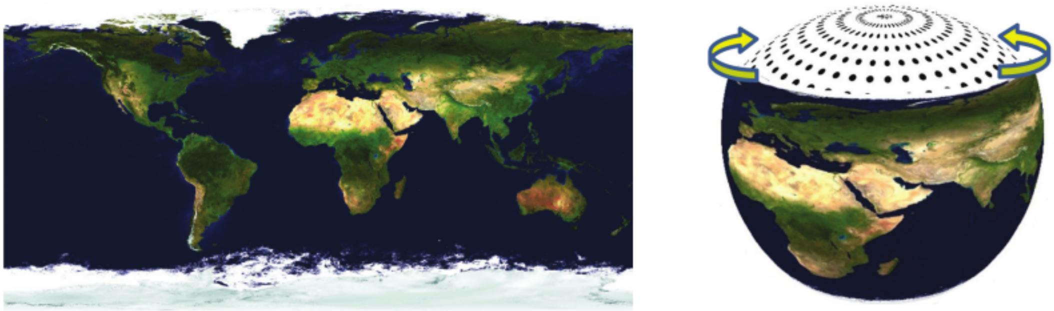
for each vertex j in slice i {

calculate indices for two triangles which point to neighboring vertices to the right, above and to the above-right of vertex

}

}

Step 6: Calculate the texture coordinate



Step 7: Assign texture coordinates to each vertex according to the resulting corresponding positions of the texels in the image

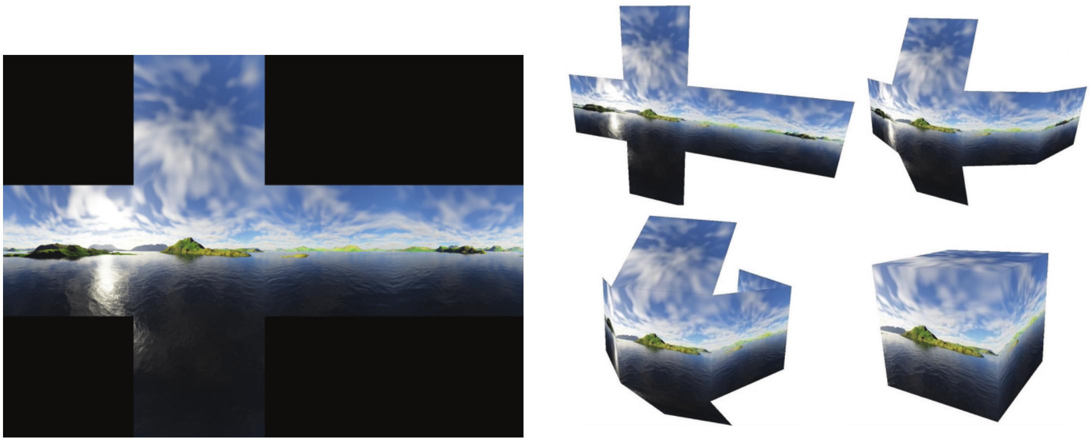
3. Matrix Stack

   Stack of transformation matrices is called matrix stack. Matrix Stack make it easy to create and manage complex hierarchical objects and scenes where transfroms can be built upon other transforms. Computing the actual path of Moon through Space is difficult. However, we can combine the transform representing the two simple circular paths - the moon's path around the Earth and the Earth's path around the Sun.

4. Skyboxes

   A skybox provides a relatively simple way of efficiently generating a convincing horizon. Procedure to create a skybox is

   Step 1: Instantiate a cube object.



   Step 2: Texture the cube with the desired environment.

   Step 3: Position the cube so it surrounds the camera.

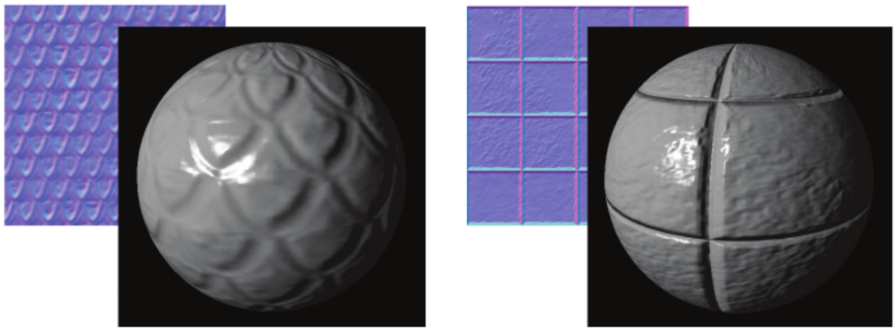
   Ways to make the skybox appear distant:

   1. Disable the depth testing and render the skybox first.
   2. Re-enable the depth testing when rendering the other objects in the scene.
   3. Move the skybox with the camera(if the camera moves)

4. By drawing the skybox first before enabling depth testing method, all the other objects will be fully rendered without being blocked by the skybox. This causes the wall of skybox to appear farther away.

5. The actual skybox cube itself can be quite small, as long as it i moved along with the camera whenever the camera moves.

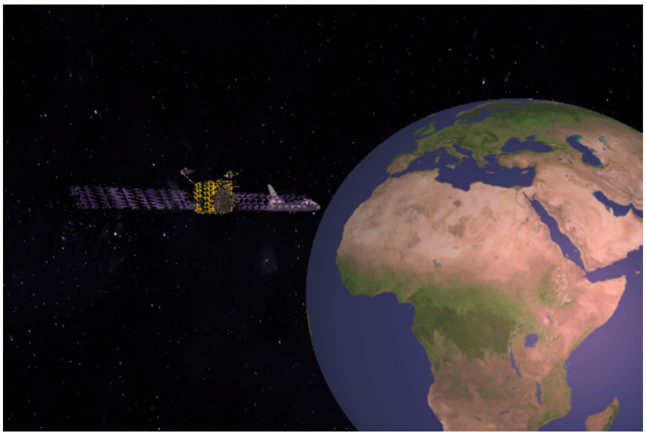
5. Normal Map

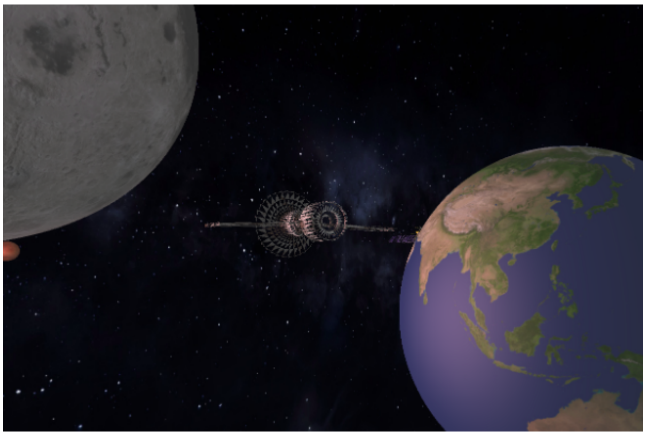


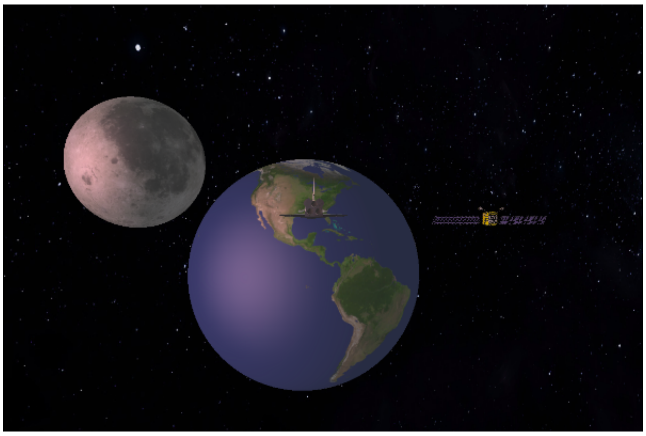
In Bump Mapping, Normal of the surface are used to produce the bumps on the objects. However, in Normal Mapping, we use lookup table which allows us to construct bumps corresponding to the craters on the object to be displayed. Normal maps are used to construct bumps for which there is no mathematical function. We used GIMP normal mapping plugin to create normal maps for our texture. Normal mapping utilizes an image file (called a normal map) that contains normals corresponding to a desired surface appearance in the presence of lightings.

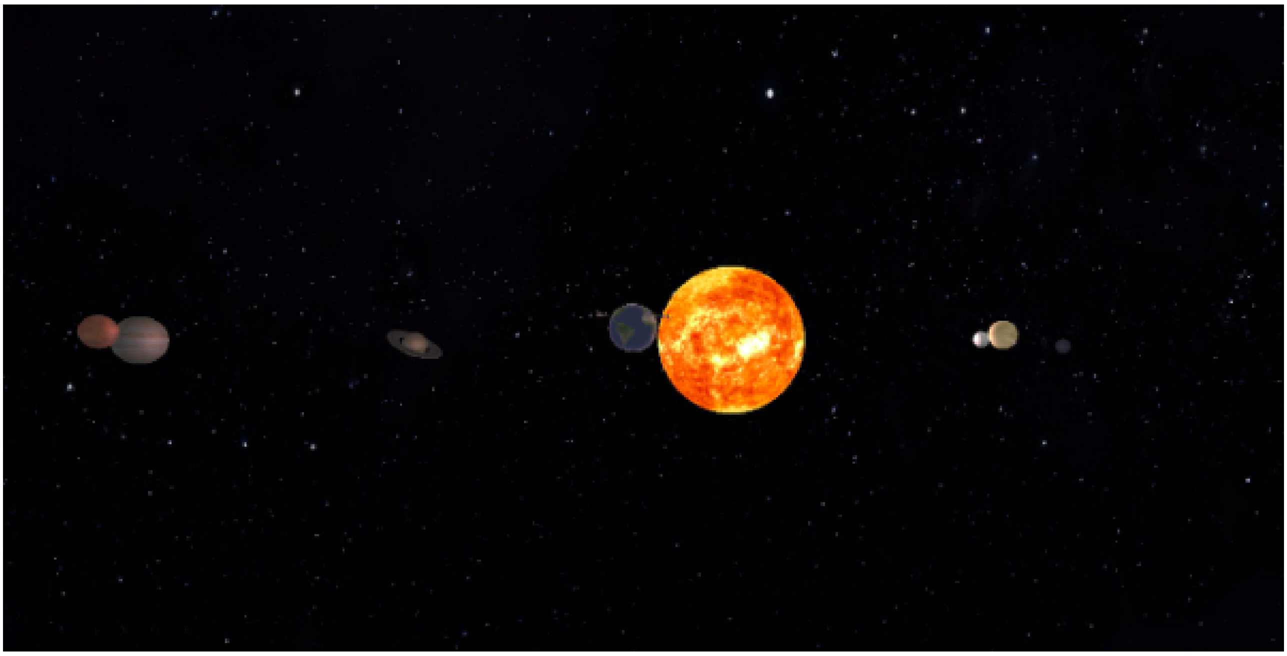
$R = \frac{N_x+1}{2}, \qquad G = \frac{N_y+1}{2}, \qquad B = \frac{N_z+1}{2}$

# Result









## Conclusion

With the successful implementation of various graphics algorithms and OpenGL C++, Gravity Simulation Project was successfully created. Surface rendering was improved with the use of normal mapping of texture image which was created using GNU image manipulation program(GIMP). Skybox and cube texture coordinates were used so as to create a convincing horizon. Matrix stack was used to calculate the position of the planets. The demo was completed by building solar system and manual addition of planets, stars and moon. 3D modelling of Satellites and Starship was added in the project as a part of 3D model rendering.

## Further Enhancement

1. Different other factors like density, axial tilt, composition, etc. can be added to simulate the rotation and revolution of heavenly bodies.
2. Variable size heavenly bodies addition button can be created
3. Shadow can be added so that we can even observe eclipse.

## References


- Gordon,V.Scott and Clevenger,John : Computer Graphics Programming in OpenGL with C++,2021
- De Vries, Joey : Learn OpenGL - Graphics programming
- Huges, John and Van Dam, Andries : Computer Graphics principles and practices
- Baker : Computer Graphics with Open GL