VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI

Mini Project Report on

"SIMULATION OF SOLAR SYSTEM"

Submitted in the partial fulfilment for the requirements of Computer Graphics & Visualization Laboratory of 6th semester CSE requirement in the form of the Mini Project work

Submitted By

Parth Chawla USN: 1BY17CS108
Ritika Shrivastava USN: 1BY17CS141

Under the guidance of

Mr. SHANKAR R
Assistant Professor, CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064

2019-2020

BMS INSTITUTE OF TECHNOLOGY &MANAGEMENT

YELAHANKA, BENGALURU – 560064

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project work entitled **"Simulation of Solar System"** is a bonafide work carried out by **Parth Chawla (1BY17CS108) and Ritika Shrivastava (1BY17CS141)** in partial fulfilment for *Mini Project* during the year 2sss019-2020. It is hereby certified that this project covers the concepts of *Computer Graphics & Visualization*. It is also certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report.

Signature of the Guide with date
Mr. SHANKAR R
Assistant Professor
CSE, BMSIT&M

Signature of HOD with date
Dr. Anil G N
Prof & Head
CSE, BMSIT&M

INSTITUTE VISION

To emerge as one of the finest technical institutions of higher learning, to develop engineering professionals who are technically competent, ethical and environment friendly for betterment of the society.

INSTITUTE MISSION

Accomplish stimulating learning environment through high quality academic instruction, innovation and industry-institute interface.

DEPARTMENT VISION

To develop technical professionals acquainted with recent trends and technologies of computer science to serve as valuable resource for the nation/society.

DEPARTMENT MISSION

Facilitating and exposing the students to various learning opportunities through dedicated academic teaching, guidance and monitoring.

PROGRAM EDUCATIONAL OBJECTIVES

1. Lead a successful career by designing, analyzing and solving various problems in the field of Computer Science & Engineering.
2. Pursue higher studies for enduring edification.
3. Exhibit professional and team building attitude along with effective communication.
4. Identify and provide solutions for sustainable environmental development.

Acknowledgement

We are happy to present this project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. We would like to express our deep sense of gratitude and indebtedness to each and every one who has helped us make this project a success.

We heartily thank our Principal, Dr. MOHAN BABU G N, BMS Institute of Technology & Management, for his constant encouragement and inspiration in taking up this project.

We heartily thank our Professor and Head of the Department, Dr. ANIL G N, Department of Computer Science and Engineering, BMS Institute of Technology &Management, for his constant encouragement and inspiration in taking up this project.

We gracefully thank our Project Guide, Mr. Shankar R, Assistant Professor, Department of Computer Science and Engineering for his intangible support and for being constant backbone for our project.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for the support and encouragement given throughout in completing this precious work successfully.

PARTH CHAWLA (1BY17CS108)

RITIKA SHRIVASTAVA (1BY17CS141)

Abstract

The Solar System consists of the Sun and the astronomical objects bound to it by gravity, all of which formed from the collapse of a giant molecular cloud approximately 4.6 billion years ago. Of the many objects that orbit the Sun, most of the mass is contained within eight relatively solitary planets whose orbits are almost circular and lie within a nearly flat disc called the ecliptic plane. The four smaller inner planets, Mercury, Venus, Earth and Mars, also called the terrestrial planets, are primarily composed of rock and metal. The four outer planets, the gas giants, are substantially more massive than the terrestrials. The two largest, Jupiter and Saturn, are composed mainly of hydrogen and helium; the two outermost planets, Uranus and Neptune, are composed largely of ices, such as water, ammonia and methane, and are often referred to separately as "ice giants".

Table of Contents

Chapter 1: Introduction

1.1 Brief Introduction

The development of technology has increased very rapidly and can be found in almost all areas of human life.

Simulation of the Solar System is modified in graphic or animation to display a collection of celestial objects that consist a large star called the sun, and all objects that are bound by the force of gravity. The objects are eight planets that make a revolution or rotation of the sun and remain in orbit respectively. The Solar System is also home to two regions populated by smaller objects. The asteroid belt, which lies between Mars and Jupiter, is similar to the terrestrial planets as Six of the planets and three of the dwarf planets are orbited by natural satellites, usually termed "moons" after Earth's Moon. Each of the outer planets is encircled by planetary rings of dust and other particles. Simulation/ Visualization of this is one using OpenGL API.

OpenGL supports this modelling capability as OpenGL has additional features to better produce something more realistic. It allows us to create a graph that can be run on any operating system only minor adjustments. OpenGL is a computer graphics rendering API or we can say that OpenGL is a library for doing computer graphics. It is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. OpenGL can used to create an interactive application that render high-quality colour images composed of 3D geometric objects and image.

Benefits of the project:

1. Used as a reference in the process of visualization of the solar system.
2. Used as a reference to learn about the solar system.

1.2 Motivation

Modelling is a representation of how people describe or explain an object, system, or a concept, which is usually manifested by simplification or idealization. This can be represented by physical models (mock-ups, prototypes), the model image (design drawings, computer images), or mathematical formulas. Simulation can also be done to facilitate the delivery of a material in a formal classroom or school.

The study of the solar system for example, by using the simulation of objects in the solar system would facilitate teachers for the delivery of content.

1.3 Scope

This project is aimed at developing a Simulation of the movement of the Celestial Bodies in our Solar System using OpenGL library which is one of the most widely used API for graphics. It was developed on CodeBlocks platform.

1.4 Problem Statement

This project aims to study and visualize the relative movement of the Planets around the Sun.

1.5 Proposed System

Computer generated models are extremely useful for teaching huge number of concepts and fundamentals in an easy to understand and learn manner. Using computer graphics many educational models can be created through which more interest can be generated among the students regarding the subject.

Today the need of visualize things have increased drastically, the need of visualization can be seen in many advance technologies, data visualization helps in finding insights of the data , to check and study the behavior of processes around us we need appropriate visualization which can be achieved through proper usage of computer graphics.

1.6 Limitations

2. The viewing angle/ perspective cannot be changed.
3. The revolution speed cannot be changed.

Chapter 2: Literature Survey

Solar System

The solar system is made up of the sun and everything that orbits around it, including planets, moons, asteroids, comets and meteoroids. It extends from the sun, called Sol by the ancient Romans, and goes past the four inner planets, through the Asteroid Belt to the four gas giants and on to the disk-shaped Kuiper Belt and far beyond to the teardrop-shaped heliopause. Scientists estimate that the edge of the solar system is about 9 billion miles (15 billion kilometers) from the sun. Beyond the heliopause lies the giant, spherical Oort Cloud, which is thought to surround the solar system.

For millennia, astronomers have followed points of light that seemed to move among the stars. The ancient Greeks named them planets, meaning "wanderers." Mercury, Venus, Mars, Jupiter and Saturn were known in antiquity, and the invention of the telescope added the Asteroid Belt, Uranus, Neptune, Pluto and many of these worlds' moons. The dawn of the space age saw dozens of probes launched to explore our system, an adventure that continues today. Only one spacecraft so far, Voyager 1, has crossed the threshold into interstellar space.

Many scientists think our solar system formed from a giant, rotating cloud of gas and dust known as the solar nebula. As the nebula collapsed because of its gravity, it spun faster and flattened into a disk. Most of the material was pulled toward the center to form the sun. Other particles within the disk collided and stuck together to form asteroid-sized objects named as planetesimals, some of which combined to become the asteroids, comets, moons and planets.

Computer Graphics

Computer graphics are graphics created using computers and, more generally, the representation and manipulation of image data by a computer hardware and software. The development of computer graphics, or simply referred to as CG, has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized the animation and video game industry. 2D computer graphics are digital

images—mostly from two-dimensional models, such as 2D geometric models, text (vector array), and 2D data. 3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering images.

OpenGL

OpenGL is the most extensively documented 3D graphics API(Application Program Interface) to date. Information regarding OpenGL is all over the Web and in print. It is impossible to exhaustively list all sources of OpenGL information. OpenGL programs are typically written in C and C++. One can also program OpenGL from Delphi (a Pascal-like language), Basic, Fortran, Ada, and other langauges. To compile and link OpenGL programs, one will need OpenGL header files. To run OpenGL programs, one may need shared or dynamically loaded OpenGL libraries, or a vendor-specific OpenGL Installable Client Driver (ICD).

GLUT

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres, and cylinders. GLUT even has some limited support for creating pop-up menus. The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. All GLUT functions start with the glut prefix (for example, glutPostRedisplay marks the current window as needing to be redrawn).

Key Stages in OpenGL Rendering Pipeline

- **Display Lists**

All data, whether it describes geometry or pixels, can be saved in a *display list* for current or later use.

- **Evaluators**

All geometric primitives are eventually described by vertices. Parametric curves and surfaces may be initially described by control points and polynomial functions called basis functions. Evaluators provide a method to derive the vertices used to represent the surface from the control points.

- **Per-Vertex Operations**

For vertex data, next is the "per-vertex operations" stage, which converts the vertices into primitives. Some vertex data (for example, spatial coordinates) are transformed by 4 x 4 floating-point matrices.

- **Primitive Assembly**

Clipping, a major part of primitive assembly, is the elimination of portions of geometry which fall outside a half-space, defined by a plane. Point clipping simply passes or rejects vertices; line or polygon clipping can add additional vertices depending upon how the line or polygon is clipped.

- **Pixel Operations**

While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route. Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step If pixel data is read from the frame buffer, pixel-transfer operations (scale, bias, mapping, and clamping) are performed. Then these results are packed into an appropriate format and returned to an array in system memory.

- **Texture Assembly**

An OpenGL application may wish to apply texture images onto geometric objects to make them look more realistic. If several texture images are used, it’s wise to put them into texture objects so that you can easily switch among them.

- **Rasterization**

Rasterization is the conversion of both geometric and pixel data into *fragments*. Each fragment square corresponds to a pixel in the framebuffer.

- **Fragment Operations**

Before values are actually stored into the framebuffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enabled or disabled. The first operation which may be encountered is texturing, where a texel (texture element) is generated from texture memory for each fragment and applied to the fragment. Then fog calculations may be applied, followed by the scissor test, the alpha test, the stencil test, and the depth-buffer test (the depth buffer is for hidden-surface removal). Failing an enabled test may end the continued processing of a fragment's square. Then, blending, dithering, logical operation, and masking by a bitmask may be performed. Finally, the thoroughly processed fragment is drawn into the appropriate buffer, where it has finally advanced to be a pixel and achieved its final resting place.

- **OpenGL-Related Libraries**

OpenGL provides a powerful but primitive set of rendering commands, and all higher-level drawing must be done in terms of these commands. Also, OpenGL programs have to use the underlying mechanisms of the windowing system.

Chapter 3: System Requirements

Hardware Requirements

- 128 MB of RAM, 256 MB recommended.

- 110 MB of hard disk space required; 40 MB additional hard disk space required for installation (150 MB total).

Software Requirements

- CodeBlocks IDE
- freeglut
- Language used: C

Chapter 4: System Analysis

We have incorporated several inbuilt OpenGL functions in this project. The following code snippet enables the easy rendering of solid sphere with different colors and makes them to rotate and translate.

```
{

. . . . .

glRotatef (. . .);
glTranslatef (. . .);
glRotatef (. . .);
glColor3f (. . .);
glutSolidSphere (. . .);
. . . . .

}
```

The header files used are:

1. #include<stdlib.h>: This is C library function for standard input and output.

2. #include<GL/glut.h>: This header is included to read the glut.h, gl.h and glu.h.

3. #include<math.h>: This is a C library function for performing certain mathematical operations.

We have made use of the following functions:

1. glClearColor(. . .):

Whenever we wish to draw a new frame, the window must be cleared by using the 4-dimensional (RGBA) colour system. The above function must be enabled to make the window on screen solid and white.

2. glshadeModel(. . .):

To enable the smooth shading, we must set the shade as follows
glShadeModel(GL_SMOOTH);

3. glEnable(. . .):

The z_buffer is one of the buffers that make up the frame buffer. The depth buffer must be cleared whenever we wish to redraw the display. This is done as follows
glEnable(GL_DEPTH_TEST);

4. glMaterial(. . .):

We can specify different material properties for the front and back faces of a surface through the following functions
glMaterialfv(GLenum face, GLenum type,GLfloat *pointer_to_array);
glMaterialfv(GLenum face,GLenum type,GLfloat value);

5. glLight(. . .):

This function is used to enable a light source.The following function specifies the required vector and scalar parameters to enable a light source.

glLightfv(GLenum source,GLenum parameter,GLfloat*pointer_to_arrary))
glLightf(GLenum source,GLenum parameter,GLfloat value)

6. glColorMaterial(. . .):

This function is used to change a single material property.

7. myinit();

Here we initialize the color buffer, set the point size and set window co-ordinate values.

8. display();

This function creates and translates all the objects in a specified locatioN in a particular order.

9. glutPostRedisplay();

It ensures that display will be drawn only once each time program goes through the event loop.

10. glutMainLoop();

This function whose execution will cause the program to begin an event processing loop.

Keyboard Based Interface

Using the keyboard user can make the planets to rotate on their own axis and revolve round the Sun. The stars are made to twinkle and the Comet is made to revolve round the Sun.

1. The keys **m, v, e, r, j, s, u, n** are used to rotate the planets.

2. The keys **M, V, E, R, J, S, U, N** are used to revolve the planets around the Sun.

3. The key **z** rotates the sun; **B** gives both the rotation and revolution of the planets around the rotating Sun with a Comet revolution and Stars twinkle.

4. Pressing the key **A** revolves all the planets and comet and the key **a** rotates all the planets around the rotating Sun with Stars twinkling in the background.

5. The key **b** is used to make the stars twinkle and **c** for the revolution of the Comet.

Chapter 5: Implementation

The implementation of the different objects in this project is divided into different module.

MODULE 1:

SUN:

The sun is drawn by using the following lines of code.

```
{
glPushMatrix();
glRotatef(…);
glLightfv(GL_LIGHT0,GL_POSITION,position);
glDisable(GL_LIGHTING);
glutSolidSphere(…);
glPopMatrix();
}
```

MODULE 2:

PLANETS WITH RINGS:

The planets Saturn and Uranus are the 2 planets in our solar system with rings. They are implemented using the following codes.

```
{
glPushMatrix();
glRotatef(…);
glTranslatef(…);
gluLookAt(0.0,10.0,2.0,1.0,0.0,0.0,0.0,0.0,1.0);
glutSolidSphere(…);

int i=0;
glBegin(GL_QUAD_STRIP);
for(i=0;i<=360;i++)
{
glVertex3f(sin(i*3.1416/180)*0.5,cos(i*3.1416/180)*0.5,0);
glVertex3f(sin(i*3.1416/180)*0.7,cos(i*3.1416/180)*0.7,0)
}
```

```
glEnd();
glPopMatrix();
}
```

MODULE 3:

EARTH:

The earth is drawn along with its natural satellite, moon which revolve round the earth. The following lines of codes are used to implement the earth and the moon.

```
{
glPushMatrix();
glRotatef(…);
glTranslatef(…);
glRotatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw planet earth*/
glRotatef(…);
glTranslatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw moon*/
glPopMatrix();
}
```

MODULE 4:

OTHER PLANETS:

The remaining planets are Mercury, Venus, Mars, Jupiter and Neptune. All these planets are implemented using the same set of codes by changing the values and colors.

```
{
glPushMatrix();
glRotatef(…);
glTranslatef(…);
glRotatef(…);
glColor3f(…);
glutSolidSphere(…); /*draw smaller planet mercury*/
glPopMatrix();
```

```
}
```

MODULE 5:

STARS:

The stars are implemented in the background using the following lines of codes.

```
{
glPushMatrix();
glTranslatef(…);
gluLookAt(0.0,10.0,0.0,1.0,0.0,0.0,0.0,0.0,3.0);
glRotatef(…);
glScalef(…);
glColor3f(…);
glutSolidSphere(…);
glPopMatrix();
}
```

MODULE 6:

COMET:

The comet is made to revolve round the sun. The following codes are used to implement the comet.

```
{
glPushMatrix();
glRotatef((GLfloat)c,6.0,-14.0,-6.0);
glTranslatef(5.0,3.0,-1.0);
glScalef(0.60,0.0,2.5);
glColor3f(7.5,9.5,2.0);
glutSolidSphere(0.2,12,6); /*draw comet*/
glPopMatrix();
}
```

Chapter 6: Conclusion

The code we have implemented for our project is working well to the best of our knowledge. In this project the planets, sun, comet and starts act as per the user's command. This project will serve as a delight to the eyes of the night sky watchers.

This project is both informative and entertaining. This project provided an opportunity to learn the various concepts of the subject in detail and provided us a platform to express our creativity and imagination come true.

Bibliography

1. Edward Angel: Interactive Computer Graphics: A Top-Down Approach with 5th Edition, Addison-wesley,2008.
2. James D Foley, Andries Van Dam, Steven K Feiner, Jhon F Hughes: - Computer Graphics, Addison-Wesley 1997.
3. Yashavanth Kanetkar," Graphics under opengl", Published by BPB Publications.
4. Athul P.Godse, "Computer Graphics", Technical Publications Pune, 2nd Revised Edition.
5. James D.foley, Andries Van Dam, Steven K.Feiner, John F.Hughes, "Computer Graphics Principle & Practice" Published by Pearson Education Pvt Ltd, 2nd Edition.