VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Computer Graphics Laboratory With Mini Project Report-18CSL67
on

"ATOM SIMULATION"

Submitted By

1BI19CS045 **CHIRANTH J S**

for the academic year 2021-22

Under the guidance of

Dr. Bhanushree K J
Assistant Professor

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K. R. Road, V. V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Graphics Laboratory With Mini Project (18CSL67)** entitled **"ATOM SIMULATION"** has been successfully completed by

1BI19CS045 **CHIRANTH J S**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2021-2022**.

Lab In charge:

Dr. Bhanushree K J
Assistant Professor
Dept. of CSE, BIT

Dr. Girija J
Professor and Head
Dept. of CSE, BIT

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my sincere thanks to **Dr. M U Aswath**, Principal, BIT and **Dr. Girija J**, HOD, Department of CS&E, BIT for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charge **Dr. Bhanushree K. J,** on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

CHIRANTH J S
1BI19CS045

Table of contents

Chapter -1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. Basically, there are 2 types of computer graphics namely,

Interactive Computer Graphics involves a two-way communication between computer and user. The observer is given some control over the image by providing him with an input device. This helps him to signal his request to the computer.

Non-Interactive Computer Graphics otherwise known as passive computer graphics it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers.

1.2 Applications of Computer Graphics

Scientific Visualization

Scientific visualization is a branch of science, concerned with the visualization of three-dimensional phenomena, such as architectural, meteorological, medical, biological systems.

Graphic Design

The term graphic design can refer to a number of artistic and professional disciplines which focus on visual communication and presentationComputer-aided Design

Computer-aided design (CAD) is the use of computer technology for the design of objects, real or virtual. The design of geometric models for object shapes, in particular, is often called computer-aided geometric design (CAGD). The manufacturing process is tied in to the computer description of the designed objects so that the fabrication of a product can be automated using methods that are referred to as CAM, computer-aided manufacturing.

Web Design

Web design is the skill of designing presentations of content usually hypertext or hypermedia that is delivered to an end-user through the World Wide Web, by way of a Web browser.

Digital Art

Digital art most commonly refers to art created on a computer in digital form.

Video Games

A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a raster display device.

Virtual Reality

Virtual reality (VR) is a technology which allows a user to interact with a computer simulated environment. The simulated environment can be similar to the real world. This allows the designer to explore various positions of an object. Animations in virtual reality environments are used to train heavy equipment operators or to analyse the effectiveness of various cabin configurations and control placements.

Computer Simulation

A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system.

Education and Training

Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behaviour. Most simulators provide screens for visual display of the external environment with multiple panels is mounted in front of the simulator.

Image Processing

The modification or interpretation of existing pictures such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image processing techniques, on the other hand, are used to improve picture quality, analyse images, or recognize visual patterns for robotics applications

1.3 OpenGL

OpenGL has become a widely accepted standard for developing graphics applications. Most of our applications will be designed to access OpenGL directly through functions in the three libraries. Functions in main GL libraries have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All function in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations. Functions in the GLU library starts with the letters glu.

The third is the OpenGL Utility Toolkit (GLUT). It provides the minimum functionality that should be formulated in modern windowing systems.

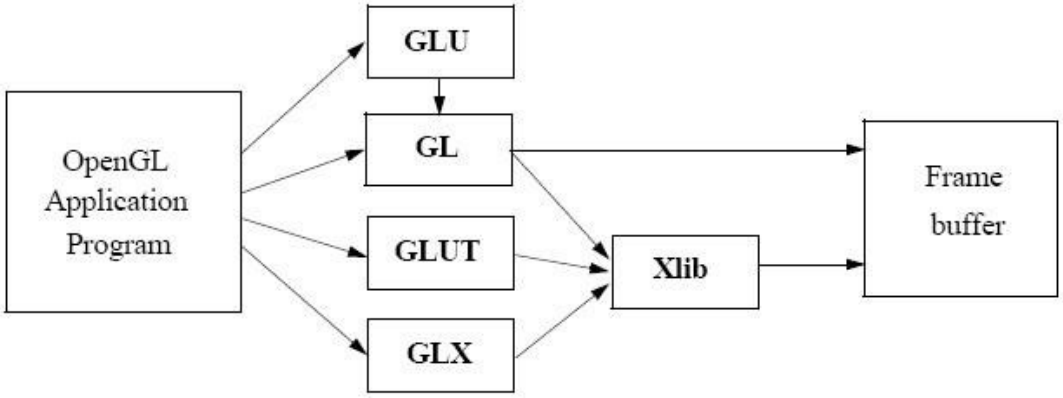

Figure 1.1 Basic block diagram of OpenGL

1.4 Problem Statement

The aim of this project is to develop 2-D atom simulator, which contains options like selecting the user desired element, simulating the selected element. And stopping the simulation when user wants. The interface should be user friendly and should use mouse and keyboard interface for the interaction with the user. The main goal is to show the users how an element structure is and how the electrons revolve around the nucleus so that one can easily get the knowledge of atom.

1.5 Objectives of the Project

- To show the working of the orthographic projections in appearance of the objects used in the scene.
- To show the implementation of Textures for a better appearance of the real-world objects.
- To show the implementation of the OpenGL transformation functions.
- To show the user and programme interaction using input devices

1.6 Organisation of the Project

The project was organised in a systematic way. First we analysed what are the basic features to be included in the project to make it acceptable. As it is a graphics oriented project, we made the sketches prior, so as to have an idea like how our output must look like. After all these, the source code was formulated as a paper work. All the required software were downloaded. Finally, the successful implementation of the project.

Chapter -2

SYSTEM SPECIFICATION

2.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

- CPU: Intel or AMD processor
- Cores: Dual-Core (Quad-Core recommended)
- RAM: minimum 4GB (>4GB recommended)
- Graphics: Intel Integrated Graphics or AMD Equivalent
- Secondary Storage: 250GB
- Display Resolution: 1366x768 (1920x1080 recommended)

2.2 Software Requirements

- Operating System: Windows 10
- Compiler: GNU C/C++ Compiler
- Development Environment: CODE BLOCKS

Chapter -3

DESIGN

3.1 Data Flow Diagram

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is part of the structured-analysis modelling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

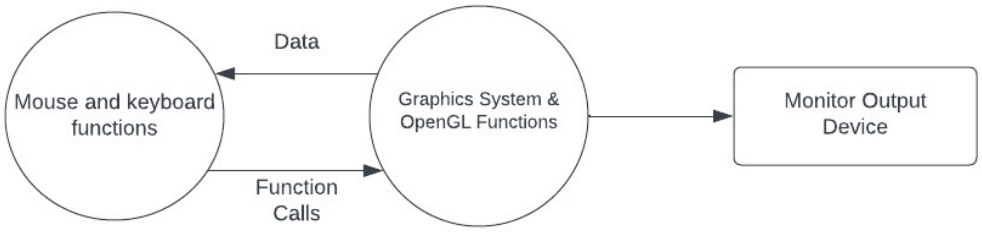

Figure 3.1 Level 0 Dataflow Diagram of the Proposed System

Figure 4.1 shows the Level 0 Dataflow diagram of the proposed application for an atom simulation where the user clicks on the enter keyboard to enter the simulation screen where the user is given options to simulate the elements from hydrogen to neon and with the help of openGL functions the atoms move around the nucleus and the output is shown on the monitor screen to the user for their desired elements.

3.2 Flowchart

A flowchart is a visual representation of the sequence of steps and decisions needed to perform a process. Each step in the sequence is noted within a diagram shape. Steps are linked by connecting lines and directional arrows.

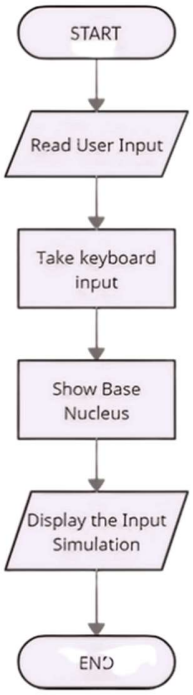

Figure 3.2 Flowchart of the Proposed System

Once the program is executed the user takes clicks on the “enter” the user is taken to the simulation screen and given a menu of elements from hydrogen to neon that is the first 10 elements of the periodic table representing their positions in the nucleus

Chapter -4

IMPLEMENTATION

4.1 Built in Functions

1. glutInit()

glutInit is used to initialize the GLUT library.

Usage: void glutInit (int *argc, char **argv);

Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. glutInitDisplayMode()

glutInitDisplayMode sets the initial display mode.

Usage: void glutInitDisplayMode (unsigned int mode);Mode-Display mode, normally the bitwise OR-ing GLUT display mode bit masks.

Description: The initial display mode is used when creating top-level windows, sub-windows, and overlays to determine the OpenGL display mode for the to-be created window or overlay.

3. glutCreateWindow()

glutCreateWindow creates a top-level window.

Usage: intglutCreateWindow (char *name); Name-ASCII character string for use as window name

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

Each created window has a unique associated OpenGL context.

4. glutDisplayFunc()

glutDisplayFunc sets the display callback for the current window. Usage: void glutDisplayFunc (void(*func)(void)); Func: The new display callback function. Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

5. glutMainLoop()

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

6. glMatrixMode()

The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of

matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

7. glTranslate(GLfloat X, GLfloat Y, GLfloat Z)

glTranslate produces a translation by x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.

8. glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)

glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.

9. glPushMatrix()

There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

10. glPopMatrix()

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix stack that contains only a single matrix. In either case, the error flag is set and no other change is made to GL state.

11. glutSwapBuffers()

Usage: void glutSwapBuffers(void);

Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefine

12. glutKeyboardFunc()

Usage: void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y)

Func**:** The new keyboard callback function

Description**:** glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

13. glLoadIdentity(void)

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.

14. void glClear(glEnum mode)

Clears the buffers namely color buffer and depth buffer. mode refers to GL_COLOR_BUFFER_BIT or DEPTH_BUFFER_BIT.

15. void glEnable(GLenum feature)

Enables an OpenGL feature. Feature can be GL_DEPTH_TEST (enables depth test for hidden surface removal), GL_LIGHTING (enables for lighting calculations), GL_LIGHTi (used to turn on the light for number i), etc.

16. void glutBitmapCharacter(void *font, int character)

Without using any display lists, glutBitmapCharacter renders the character in the named bitmap font. The fonts used are GLUT_BITMAP_TIMES_ROMAN_24.

17. void glViewport(int x ,int y, GLsizei width, GLsizei height)

Specifies the width*height viewport in pixels whose lower left corner is at (x,y) measured from the origin of the window.

18. void glRasterPos[234][ifd](GLfloat x, GLfloat y, GLfloat z);

This position, called the raster position, is used to position pixel and bitmap write operations.

19. void glutReshapeFunc(void *f(int width, int height));

Registers the reshape callback function f. the callback function returns the height and width of the new window. The reshape callback invokes the display

callback.

4.2 User Defined Functions

- void circle(float rad)
- void drawString(float x, float y, float z, char* string

- void drawhead(float x, float y, float z, char* string)
- void drawsubhead(float x, float y, float z, char* string)
- void nuc(float rad)
- void eleright(float rad)
- void eleleft(float rad)
- void eletop(float rad)
- void eledown(float rad)
- void eletr(float rad)
- void eletl(float rad)
- void eledl(float rad)
- void eledr(float rad)
- void rotate()
- void mouseControl(int button, int state, int x, int y)
- void keyboard(unsigned char key, int x, int y)
- void fkey(int key, int x, int y)

- void menu(int option)

- void createMenu(void)

4.3 CODE

```
#include<windows.h>
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>
#include <dos.h>
#include<stdlib.h>
#include<GL/glu.h>
#include<GL/gl.h>
#define pi 3.142
static GLfloat angle = 0;
static int submenu;
static int mainmenu;
static int value = -1;
void displayRasterText(float x ,float y ,float z ,char *stringToDisplay) {
        glRasterPos3f(x, y, z);
        for(char* c = stringToDisplay; *c != '\0'; c++){
                glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24 , *c);
        }
}
void init()
{
   gluOrtho2D(-1000, 1000, -1000, 1000);
}
void circle(float rad)
```

```
{
   glBegin(GL_POINTS);
   glColor3f(1, 0, 0);
   for (float i = 0; i < (2 * pi); i += 0.00001)
   {
      glVertex2i(rad * cos(i), rad * sin(i));
   }
   glEnd();
}
void drawString(float x, float y, float z, char* string)
{
   //text colour
   glColor3f(0, 0, 0);
   //glColor3f(1, 1, 1);
   glRasterPos3f(x, y, z);

   for (char* c = string; *c != '\0'; c++)
   {
      glutBitmapCharacter(GLUT_BITMAP_HELVETICA_10, *c);    }
}

void drawhead(float x, float y, float z, char* string)
{
   //text color
   glColor3f(1, 1, 1);
   //glColor3f(1, 1, 1);
   glRasterPos3f(x, y, z);

   for (char* c = string; *c != '\0'; c++)
   {
      glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, *c);
```

```
    }
}
void drawsubhead(float x, float y, float z, char* string)
{
    //text color
    glColor3f(0, 0, 0);
    //glColor3f(1, 1, 1);
    glRasterPos3f(x, y, z);
    for (char* c = string; *c != '\0'; c++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_12, *c);
    }
}
void nuc(float rad)
{
    glBegin(GL_POLYGON);
    //polygon colour of nucli
    glColor3f(0.8, 0.4, 0.5);
    //glColor3f(0.8, 0.4, 0.5);
    for (float i = 0; i < (2 * pi); i = i + 0.00001)
    {
        glVertex2f(rad * cos(i), rad * sin(i));
    }
    glEnd();
}
void eleright(float rad)
{
    glBegin(GL_POLYGON);
    //electron colour
    glColor3f(0, 0, 1);
    //glColor3f(1, 1, 1);
```

```
    for (float i = 0; i < (2 * pi); i += 0.00001)
    {
        glVertex2i(rad + 20 * cos(i), 20 * sin(i));
    }
    glEnd();
}
void eleleft(float rad)
{
    glBegin(GL_POLYGON);
    //electron colour
    glColor3f(0, 0, 1);
    //glColor3f(1, 1, 1);
    for (float i = 0; i < (2 * pi); i += 0.00001)
    {
        glVertex2i(-(rad + 20 * cos(i)), 20 * sin(i));
    }
    glEnd();
}
void eletop(float rad)
{
    glBegin(GL_POLYGON);
    //electron colour
    glColor3f(0, 0, 1);
    //glColor3f(1, 1, 1);
    for (float i = 0; i < (2 * pi); i += 0.00001)
    {
        glVertex2i(20 * cos(i), rad + 20 * sin(i));
    }
    glEnd();
}
void eledown(float rad)
```

```
{
   glBegin(GL_POLYGON);
   //electron colour
   glColor3f(0, 0, 1);
   //glColor3f(1, 1, 1);
   for (float i = 0; i < (2 * pi); i += 0.00001)
   {
      glVertex2i(20 * cos(i), -(rad + 20 * sin(i)));
   }
   glEnd();
}
void eletr(float rad)
{
   glBegin(GL_POLYGON);
   //electron colour
   glColor3f(0, 0, 1);
   //glColor3f(1, 1, 1);
   for (float i = 0; i < (2 * pi); i += 0.00001)
   {
      glVertex2i(((rad - 175) + 20 * cos(i)), ((rad - 175) + 20 * sin(i)));
   }
   glEnd();
}
void eletl(float rad)
{
   glBegin(GL_POLYGON);
   //electron colour
   glColor3f(0, 0, 1);
   //glColor3f(1, 1, 1);
   for (float i = 0; i < (2 * pi); i += 0.00001)
   {
```

```
        glVertex2i(-((rad - 175) + 20 * cos(i)), ((rad - 175) + 20 * sin(i)));
    }
    glEnd();
}
void eledl(float rad)
{
    glBegin(GL_POLYGON);
    //electron colour
    glColor3f(0, 0, 1);
    //glColor3f(1, 1, 1);
    for (float i = 0; i < (2 * pi); i += 0.00001)
    {
        glVertex2i(-((rad - 175) + 20 * cos(i)), -((rad - 175) + 20 * sin(i)));
    }
    glEnd();
}
void eledr(float rad)
{
    glBegin(GL_POLYGON);
    //electron colour
    glColor3f(0, 0, 1);
    //glColor3f(1, 1, 1);
    for (float i = 0; i < (2 * pi); i += 0.00001)
    {
        glVertex2i(((rad - 175) + 20 * cos(i)), -((rad - 175) + 20 * sin(i)));
    }
    glEnd();
}
void display()
{
    glClearColor(0.0, 0.0, 0.0, 1);
```

```
   glClear(GL_COLOR_BUFFER_BIT);

   if (value == -1)
   {
      /*char prn[] = "A Mini Project On";
      drawsubhead(-150, 450, 0, prn);
      char pro[] = "ATOM SIMULATION";
      drawhead(-250, 350, 0, pro);

      char in[] = "Press enter to Continue";
      drawhead(-250, -700, 0, in);

      glutDetachMenu(GLUT_RIGHT_BUTTON);
      glutSwapBuffers();*/
      glClear(GL_COLOR_BUFFER_BIT);

                  glColor3f(1.0, 0.0, 0.0);
         displayRasterText(-625, 490, 0.0,"BANGALORE INSTITUTE OF
TECHNOLOGY");
                  glColor3f(0.0, 1.0, 1.0);
         displayRasterText(-975, 385, 0.0,"DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING");
                  glColor3f(0.0, 0.0, 1.0);
         displayRasterText(-225, 300, 0.0,"A MINI PROJECT ON ");
                  glColor3f(1.0, 0.0, 1.0);
         displayRasterText(-125, 225, 0.0,"Atom Simulation");
                  glColor3f(1.0, 0.7, 0.8);
         displayRasterText(-100, 150, 0.0,"created by");
                  //glColor3f(1.0, 1.0, 1.0);
          //displayRasterText(-130, 80, 0.0,"SHOOTERS");
                  glColor3f(1.0, 0.0, 0.0);
```

```
          displayRasterText(-950, -100, 0.0," STUDENTS");
                glColor3f(1.0, 0.0, 1.0);
        displayRasterText(-950, -200, 0.0," BHUVAN S GOWDA");
        displayRasterText(-950, -285, 0.0," CHIRANTH J S");
                glColor3f(1.0, 0.0, 0.0);
        displayRasterText(150, -100, 0.0,"Under the Guidance of");
                glColor3f(1.0, 0.0, 1.0);
        displayRasterText(150, -200, 0.0,"Prof BHANUSHREE K J");
                glColor3f(1.0, 0.0, 0.0);
        displayRasterText(-300, -400, 0.0,"Academic Year 2021-2022");
    glColor3f(1.0, 0.0, 1.0);
        displayRasterText(-350, -550, 0.0,"Press ENTER to start the game");
        glFlush();
        glutSwapBuffers();
}
if (value != -1)
{
    nuc(250);
    char n[] = "NUCLEUS";
    drawString(-90, 20, 0, n);
    char d[] = "(NEUTRON + PROTON)";
    drawString(-170, -30, 0, d);
    if (value == 0)
    {
        char nu[] = "SELECT THE ELEMENT USING MENU";
        drawhead(-490, 900, 0, nu);
        glutSwapBuffers();
    }

}
if (value == 1)
```

```
{
    char n[] = "HYDROGEN";
    drawhead(-100, 900, 0, n);
    circle(400);
    char o[] = "ORBIT";
    drawString(410, 0, 0, o);
    glPushMatrix();
    glRotatef(angle, 0, 0, 1);
    eleright(400);
    char e[] = "ELECTRON";
    drawString(420, 0, 0, e);
    glPopMatrix();
    glutSwapBuffers();
}
if (value == 2)
{
    char n[] = "HELIUM";
    drawhead(-100, 900, 0, n);
    circle(400);
    char o[] = "ORBIT";
    drawString(410, 0, 0, o);
    glPushMatrix();
    glRotatef(angle, 0, 0, 1);
    eleright(400);
    eleleft(400);
    char e[] = "ELECTRON";
    drawString(420, 0, 0, e);
    glPopMatrix();
    glutSwapBuffers();
}
```

```
if (value == 3)
  {
     char n[] = "LITHIUM";
     drawhead(-100, 900, 0, n);
     circle(400);
     circle(600);
     char o[] = "ORBIT";
     drawString(610, 0, 0, o);
     glPushMatrix();
     glRotatef(angle, 0, 0, 1);
     eleright(400);
     eleleft(400);
     eletop(600);
     char e[] = "ELECTRON";
     drawString(0, 620, 0, e);
     glPopMatrix();
     glutSwapBuffers();
  }
  if (value == 4)
  {
     char n[] = "BERYLLIUM";
     drawhead(-100, 900, 0, n);
     circle(400);
     circle(600);
     char o[] = "ORBIT";
     drawString(610, 0, 0, o);
     glPushMatrix();
     glRotatef(angle, 0, 0, 1);
     eleright(400);
     eleleft(400);
     eletop(600);
```

```
    eledown(600);
    char e[] = "ELECTRON";
    drawString(0, 620, 0, e);
    glPopMatrix();
    glutSwapBuffers();
}
if (value == 5)
{
    char n[] = "BORON";
    drawhead(-100, 900, 0, n);
    circle(400);
    circle(600);
    char o[] = "ORBIT";
    drawString(610, 0, 0, o);
    glPushMatrix();
    glRotatef(angle, 0, 0, 1);
    eleright(400);
    eleleft(400);
    eletop(600);
    eledown(600);
    eletr(600);
    char e[] = "ELECTRON";
    drawString(0, 620, 0, e);
    glPopMatrix();
    glutSwapBuffers();
}
if (value == 6)
{
    char n[] = "CARBON";
    drawhead(-100, 900, 0, n);
    circle(400);
```

```
        circle(600);
        char o[] = "ORBIT";
        drawString(610, 0, 0, o);
        glPushMatrix();
        glRotatef(angle, 0, 0, 1);
        eleright(400);
        eleleft(400);
        eletop(600);
        eledown(600);
        eletr(600);
        eledl(600);
        char e[] = "ELECTRON";
        drawString(0, 620, 0, e);
        glPopMatrix();
        glutSwapBuffers();
    }
    if (value == 7)
    {
        char n[] = "NITROGEN";
        drawhead(-100, 900, 0, n);
        circle(400);
        circle(600);
        char o[] = "ORBIT";
        drawString(610, 0, 0, o);
        glPushMatrix();
        glRotatef(angle, 0, 0, 1);
        eleright(400);
        eleleft(400);
        eletop(600);
        eledown(600);
        eletr(600);
```

```
    eledl(600);
    eletl(600);
    char e[] = "ELECTRON";
    drawString(0, 620, 0, e);
    glPopMatrix();
    glutSwapBuffers();
  }
  if (value == 8)
  {
    char n[] = "OXYGEN";
    drawhead(-100, 900, 0, n);
    circle(400);
    circle(600);
    char o[] = "ORBIT";
    drawString(610, 0, 0, o);
    glPushMatrix();
    glRotatef(angle, 0, 0, 1);
    eleright(400);
    eleleft(400);
    eletop(600);
    eledown(600);
    eletr(600);
    eledl(600);
    eletl(600);
    eledr(600);
    char e[] = "ELECTRON";
    drawString(0, 620, 0, e);
    glPopMatrix();
    glutSwapBuffers();
  }
  if (value == 9)
```

```
{
    char n[] = "FLUORINE";
    drawhead(-100, 900, 0, n);
    circle(400);
    circle(600);
    char o[] = "ORBIT";
    drawString(610, 0, 0, o);
    glPushMatrix();
    glRotatef(angle, 0, 0, 1);
    eleright(400);
    eleleft(400);
    eletop(600);
    eledown(600);
    eletr(600);
    eledl(600);
    eletl(600);
    eledr(600);
    eleleft(600);
    char e[] = "ELECTRON";
    drawString(0, 620, 0, e);
    glPopMatrix();
    glutSwapBuffers();
}
if (value == 10)
{
    char n[] = "NEON";
    drawhead(-100, 900, 0, n);
    circle(400);
    circle(600);
    char o[] = "ORBIT";
    drawString(610, 0, 0, o);
```

```
        glPushMatrix();
        glRotatef(angle, 0, 0, 1);
        eleright(400);
        eleleft(400);
        eletop(600);
        eledown(600);
        eletr(600);
        eledl(600);
        eletl(600);
        eledr(600);
        eleleft(600);
        eleright(600);
        char e[] = "ELECTRON";
        drawString(0, 620, 0, e);
        glPopMatrix();
        glutSwapBuffers();
    }

    //glutSwapBuffers();
}
void rotate()
{
    angle = angle + 30.0;
    if (angle > 360)
    {
        angle = angle - 360;
    }
    glClear(GL_COLOR_BUFFER_BIT);
    glutPostRedisplay();
}
void mouseControl(int button, int state, int x, int y)
```

```
{
   switch (button)
   {
   case GLUT_LEFT_BUTTON:
      if (state == GLUT_DOWN)
         glutIdleFunc(rotate);
      break;
   default:
      break;
   }
}
void keyboard(unsigned char key, int x, int y)
{
   if (key == 13)
   {
      value = 0;
      glClear(GL_COLOR_BUFFER_BIT);
      glutAttachMenu(GLUT_RIGHT_BUTTON);
      glutPostRedisplay();
   }
   else if (key == 's')
   {
      glutIdleFunc(NULL);
   }
   else if (key == 32)
   {
      glutIdleFunc(rotate);
   }
   else if (key == 'q' || key == 'Q')
   {
      exit(0);
```

```
    }
    else if (key == 'b' || key == 'B')
    {
        value = -1;
    }
    else if (key == 27)
    {
        glutReshapeWindow(700, 700);
    }

}
void fkey(int key, int x, int y)
{

    if (key == GLUT_KEY_F10)
    {
        glutReshapeWindow(glutGet(GLUT_SCREEN_WIDTH),
glutGet(GLUT_SCREEN_HEIGHT));
    }
}
void menu(int option)
{
    if (option == 13)
    {
        exit(0);
    }
    else if (option == 11)
    {
        glutIdleFunc(rotate);
    }
    else if (option == 12)
```

```
    {
        glutIdleFunc(NULL);
    }
    else if (option == 14) {
        value = -1;
    }
    else
    {
        value = option;
    }
    glClear(GL_COLOR_BUFFER_BIT);

    glutPostRedisplay();
}
void createMenu(void)
{
    submenu = glutCreateMenu(menu);
    glutAddMenuEntry("HYDROGEN", 1);
    glutAddMenuEntry("HELIUM", 2);
    glutAddMenuEntry("LITHIUM", 3);
    glutAddMenuEntry("BERILIUM", 4);
    glutAddMenuEntry("BORON", 5);
    glutAddMenuEntry("CARBON", 6);
    glutAddMenuEntry("NITROGEN", 7);
    glutAddMenuEntry("OXYGEN", 8);
    glutAddMenuEntry("FLUORINE", 9);
    glutAddMenuEntry("NEON", 10);
    mainmenu = glutCreateMenu(menu);
    glutAddSubMenu("SELECT THE ELEMENT", submenu);
    glutAddMenuEntry("START SIMULATION", 11);
    glutAddMenuEntry("STOP SIMULATION", 12);
```

```
    glutAddMenuEntry("GOTO HOME SCREEN", 14);
    glutAddMenuEntry("EXIT", 13);

    glutAttachMenu(GLUT_RIGHT_BUTTON);
}
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitWindowPosition(300, 100);
    glutInitWindowSize(700, 700);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutCreateWindow("ATOM SIMULATION");
    init();
    glutDisplayFunc(display);
    glutMouseFunc(mouseControl);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(fkey);
    createMenu();
    glutMainLoop();
    return 0;
}
```

Chapter -5

SNAPSHOTS

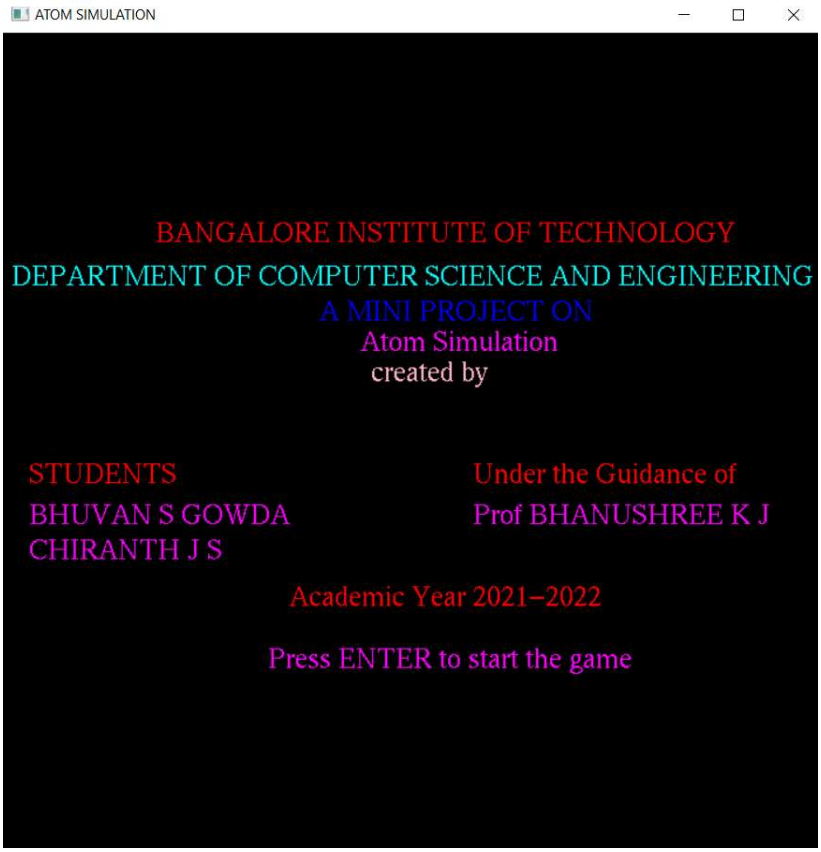

Figure 5.1 : Intro scene

Figure 5.2 : Scene 1

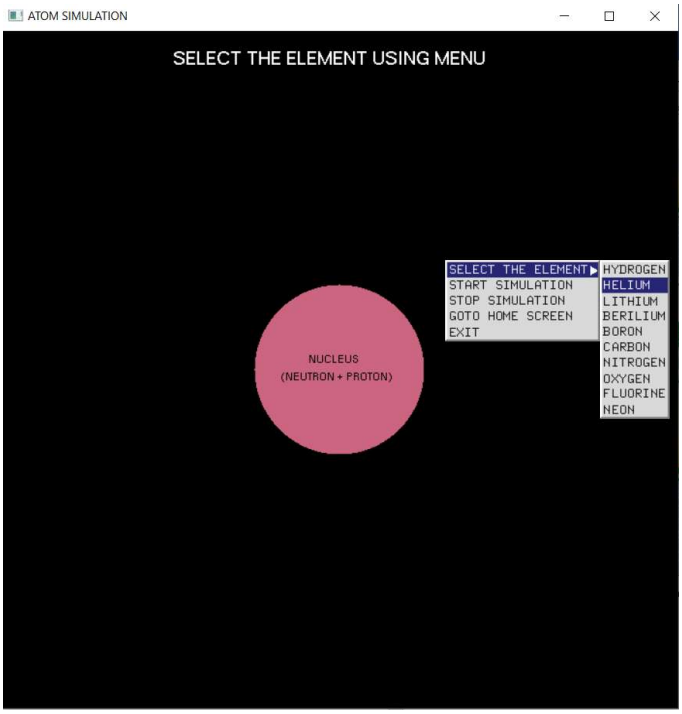

Figure 5.3 : Scene 2

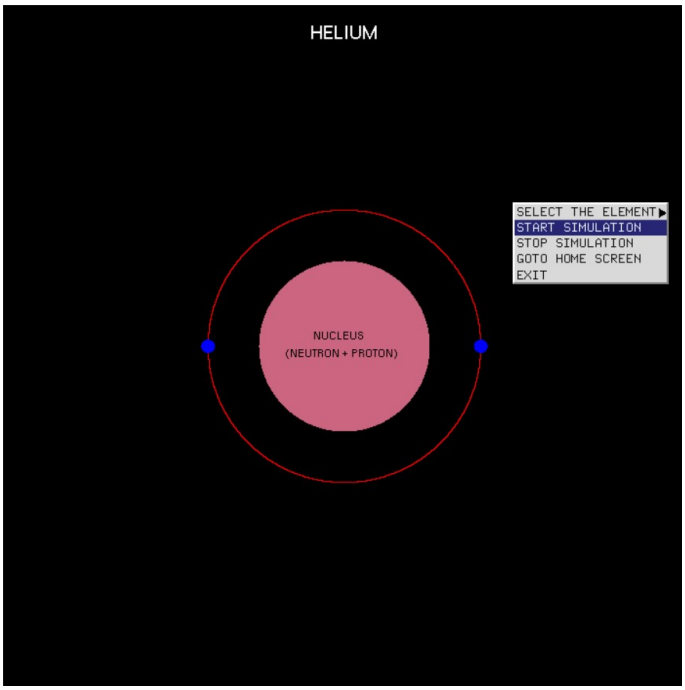

Figure 5.4 : Scene 3

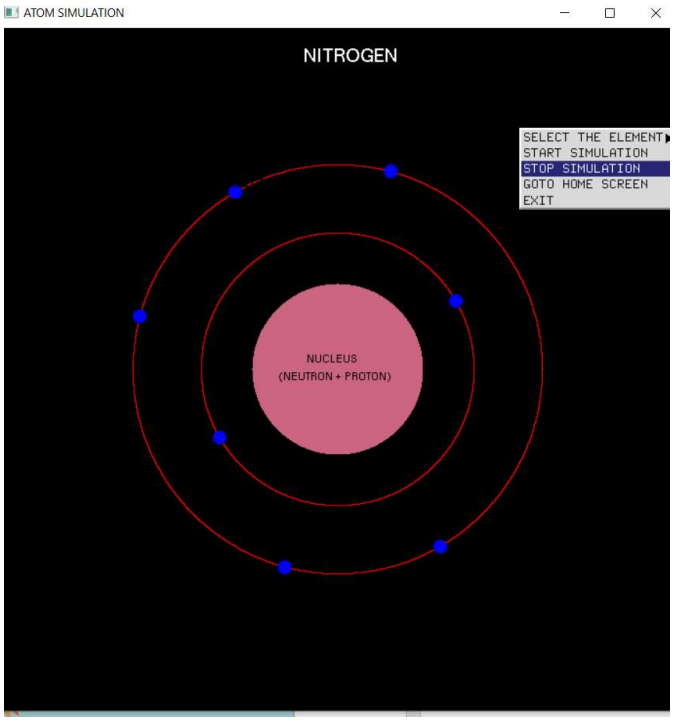

Figure 5.5 : Scene 4

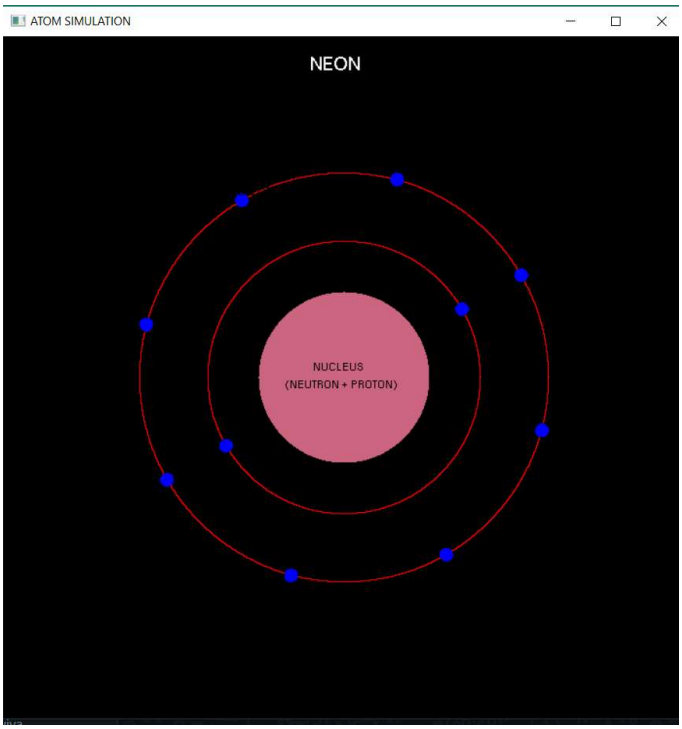

Figure 5.6 : Scene 5

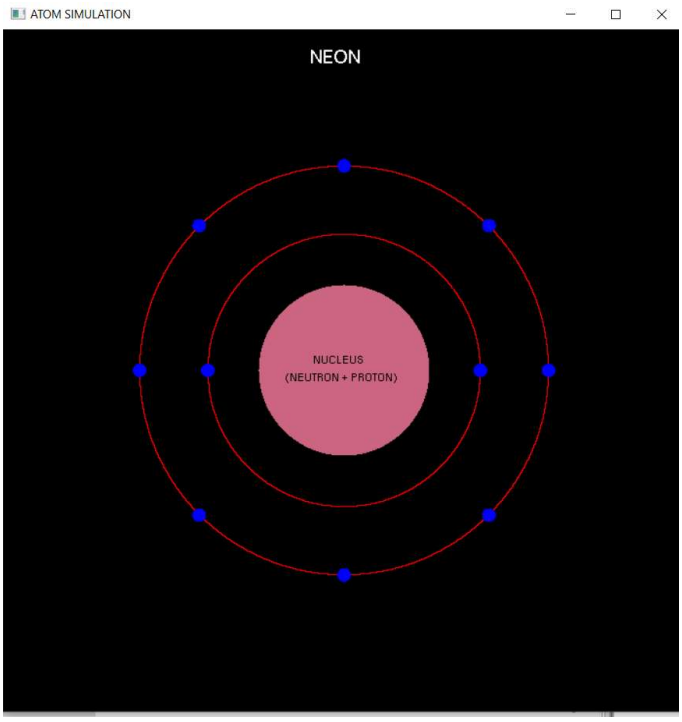

Figure 5.7 : Scene 6

Chapter -6

CONCLUSION

This atom simulation is very good project. Users can very easily understand the structure of an element. The interface is mouse driven and the user can select a function by clicking. And the interface supports keyboard interface. We have tried our best to make this simulator very realistic, so that user can easily understand the concepts of electrons, orbits, atoms and nucleus etc.

6.1 Future Enhancements

The following are some of the features that are planned to be supported in the future versions of the atom simulator.

- Adding all the elements from the periodic table.
- Features like showing the simulation with all the important details of an element.
- Adding a search bar for selecting an element from the list of all the elements.
- Making the simulation in 3-D.

6.2 Bibliography

1. Edward Angel: Interactive Computer Graphics: A Top Down Approach 5th Edition, Addison – Wesley, 2008
2. Computer Graphics Using OpenGL – F.S. Hill Jr. 2nd Edition, Pearson Education, 2001.
3. Computer Graphics – James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Addison-Wesley 1997.
4. Computer Graphics - OpenGL Version – Donald Hearn and
5. Pauline Baker, 2nd Edition, Pearson Education, 2003.
6. www.google.com
7. www.openglforum.org