VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum-590018.

Computer Graphics mini Project On

"Atom Simulation"

Submitted in partial fulfillment for the requirements 0f the VI Semester degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

For The Academic Year
2021-22
By

Latharani (1DB20CS405)

Under the Guidance Of

Prof .Vishesh J
Asst Professor
Dept. of CSE

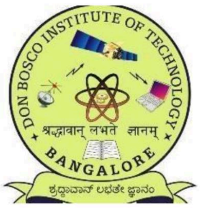
Department of Computer Science and Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560 074.

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysore Road, Bengaluru - 560074.



ACKNOWLEDGEMENT

Here by I am submitting the CG mini project report on **"Atom Simulation"**, as per the scheme of Visvesvaraya Technological University, Belgaum.

In this connection, I would like to express my deep sense of gratitude to my beloved institution Don Bosco Institute of Engineering and also, I like to express my sincere gratitude and indebtedness to **Prof Umashankar B.S , Principal, DBIT, Bangalore.**

I would like to express my sincere gratitude to **Dr K.B Shivakumar** Professor and Head of Dept. of Computer Science and Engineering, for providing a congenial environment to work in and carryout my mini project.

I would like to express the deepest sense of gratitude to thank my Project Guide **Prof Vishesh J** Asst Professor, Department of Computer Science and Engineering, DBIT, Bangalore for his constant help and support extended towards me during the course of the project.

Finally I am very much thankful to all the teaching and non teaching members of the Department of Computer Science and Engineering, my seniors, friends and any parents for their constant encouragement, support and help throughout completion of mini project.

Latharani

(1DB20cs405)

Kumbalagodu, Bengaluru – 560 074.

DEPARTMENT OF COMPUTER SCINECE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project report entitled **"Atom Simulation"** is a bonafide work carried out by **Latharani (1DB20CS405)** in partial fulfillment of award of Degree of **Bachelor of Engineering in Computer Science and Engineering** of Visvesvaraya Technological University, Belagavi, during the academic year 2021-2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The mini project has been approved as it satisfies the academic requirements associated with the degree mentioned.

Signature of guide

……

Prof. Vishesh J
Assistant Prof.
Dept. of CSE,
DBIT, Bengaluru.

.

Signature of HOD

……..

Dr. K B Shivakumar
Head of Dept.,
Dept. of CSE,
DBIT, Bengaluru.

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Bangalore – 560074.

DECLARATION

I, am Latharani, student of sixth semester B.E, Department of Computer Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bangalore, declare that the mini project work entitled **"Atom Simulation"** has been carried out by me and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during the academic year **2021-22.** The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bangalore **Latharani**
Date:07/07/2022 **1DB20CS405**

ABSTRACT

Everything you see around you is made up of atoms, and all atoms consist of subatomic particles. In the Atom simulation, you will learn the names of the basic subatomic particles and understand.

As a part of the project, you'll see how the electrons are revolving around the nucleus in their respective orbits. One can see and spot the nucleus, atoms and electrons and can understand how an electron revolves around the nucleus. The project has made in such a way that one can easily understand the simulation of atoms.

This project has been developed in Windows OS with interfacing keyboard and mouse with menu driven interface. And plans to include lighting, shading and other features in future enhancement.

This project is written in C and used OpenGL (Open Graphics Library). Open Graphics Library is a cross-language, cross-platform application programming interface for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit, to achieve hardware-accelerated rendering.

CONTENTS

Acknowledgement	**(I)**
Abstract	**(II)**
List of Figures	**(III)**
List of Abbreviations	**(III)**

CHAPTERS				Pg. No
1.			**INTRODUCTION**	**1**
2.			**LITERATURE SURVEY**	**2**
3.			**PROPOSED SYSTEM**	**3**
4.			**ANALYSIS**	4
	4.1		Software Requirements	4
	4.2		Hardware Requirements	4
5.			**SYSTEM DESIGN**	5
	5.1		FLOWCHART/ARCHITECTURE	5
	5.2		CONCEPTS AND PRINCIPLES OF OPENGL	6
	5.3		OPENGL RELATED LIBRARIES	7
	5.4		WINDOWS MANAGEMENT IN OPENGL	7
	5.5		CALLBACK FUNCTIONS	8
	5.6		RUNNING THE PROGRAM	11
6.			**IMPLEMENTION**	**12-16**
			Overview of System Implementation	16
7.			**SNAPSHOTS**	**17**
			CONCLUSION	**20**
			REFERENCES	**21**

CHAPTER 1

INTRODUCTION

• Introduction to Computer Graphics

Computer graphics started with the display of data on hardcopy plotters and Cathode Ray Tube (CRT) screens soon after the Introduction of computers themselves. It has grown to include the Creation, Storage and Manipulation of Models and Images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural and even conceptual structures, natural phenomenon and so on. Computer graphics today is largely interactive: the user controls the contents, structure, and appearance of objects and of their displayed images by using input devices, such as a keyboard, mouse, or touch-sensitive panel on the screen. Because of the close relationship between the input devices and the display, the handling of such devices is included in the study of computer graphics.

The goal of a The aim of this project is to develop a 2-D atom simulator, which contains options like selecting the user desired element, simulating the selected element. And also 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 revolves around the nucleus so that one can easily get the knowledge of atom.

CHAPTER 2

LITERATURE SURVEY

OpenGL, a graphics software system has become a widely accepted standard for developing graphics applications. Fortunately, OpenGL is easy to learn and it possesses most of the characteristics of other popular graphics system.

- Using OpenGL library, we can create various graphics related structure.
- Different functions defined in the OpenGL can be used to give different colors and textures to the ghost and Pacman.
- Glut (Toolkit of OpenGL library) can be used to create window, translate and rotate the matrix of the object, to track the position of mouse, to show the render screen etc.

Example:

- To create the window Function used: glutCreateWindow ("window name")
- To define the position of window Function used: glutInitWindowposition (int x, int y) X and Y are the co-ordinates of top left part of window in pixel.
- Define the size of window Function used: glutInitWindowsize (int x, int y) x and y arethe width and height of the window.

Likewise, many user-defined functions are used to manipulate the game flow. The works are commented in the source code. In this way I gathered most of my knowledge from the previously made similar projects and got familiar with Open GL library and its toolki ``

CHAPTER 3

PROPOSED SYSTEM

In the proposed system, OpenGL is a graphic software system designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, we must work through whatever windowing system controls the hardware you're using.

CHAPTER 4

ANALYSIS

Visual Studio 2005 delivers on Microsoft's vision of smart client applications by letting developers quickly create connected applications that deliver the highest quality rich user experiences. This new version lets any size organization create more secure, more manageable, and more reliable applications that take advantage of Windows Vista, windows7, 2007 Office System and the Web. By building these new types of applications, organizations will find it easier than ever to capture and analyze information so that they can make effective business decisions.

5.1 Software Requirements

- Operating System: Windows 10
- Compiler: Microsoft Visual Studio 10
- OpenGL: library Files And dll Files
- Editor: Visual C++
- Language: C++

5.2 Hardware Requirements

- Processor: Intel Core i3
- RAM: 8GB
- Hard Disk: 20GB
- Keyboard: Standard qwerty serial or PS/2 keyboard
- Mouse: Standard serial or PS2 mouse
- Monitor: VGA color Monitor

CHAPTER 5

5.1 FLOWCHART/ARCHECTURE

The system is designed with the maze transmission with the following system design description. The program starts with start with the main function. There are init, display, keyboard function. Then the control flows to the init function then the control flows to display function then followed by keyboard function these are used to operate step by step process of transmission

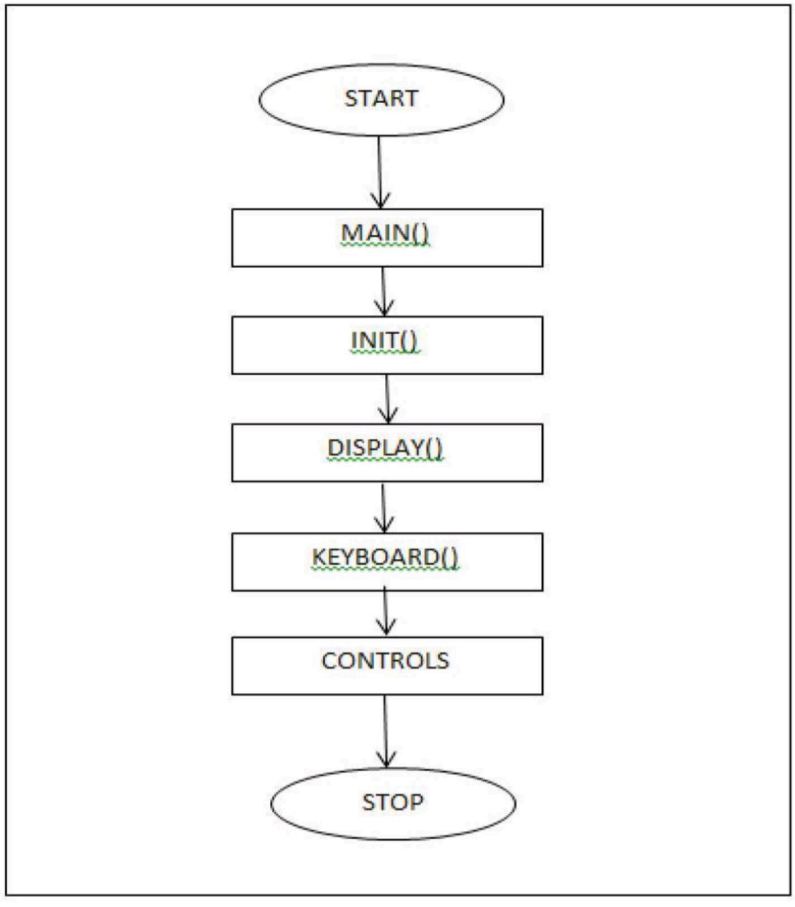


Figure 5.1: Flow Diagram

5.1 CONCEPTS AND PRINCIPLES OF OPENGL

5.2.1 THE OPENGL MODEL:

Figure relationships between an application program, the graphics system, input and output devices, and the user.

5.2.2 THE GRAPHICAL INTERFACE:

The application program has its own internal model. It draws the graphics using the facilities of the graphics system. The user views the graphics, and uses input devices, such as a mouse, to interact. Information about the users is sent back to the applications are sent back to the application, which decides what action to take. Typically, it will make changes to its internal model, which will cause the graphics to be updated, and so another loop in the interaction cycle begins.

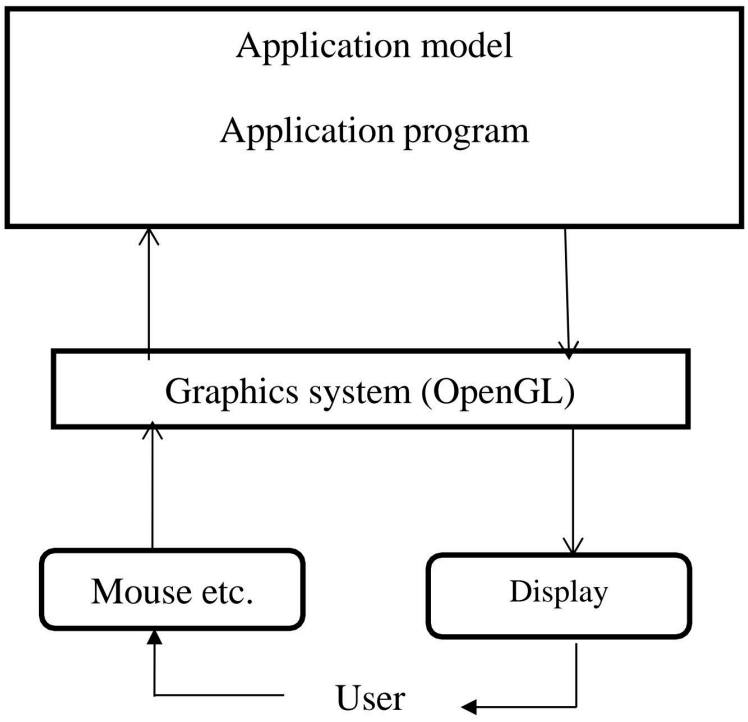


Fig 5.2: Graphical interaction loop

5.3 OPENGL related libraries

5.3.1 LIBRARIES:

OpenGL provides a powerful but primitive set of rendering command, and all the higher-level drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

- OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.
- OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kilgard, to hide the complexities of differing window APIs.

5.3.2 INCLUDE FILES:

For all OpenGL applications, you want to include the gl.h header file in every file. Almost all OpenGL applications use GLU, which also requires inclusion of the glu.h header file. So almost every OpenGL source file begins with:

#include<GL/gl.h>; #include<GL/glut.h>

5.4 WINDOWS MANAGEMENT IN OPENGL

Five routines perform tasks necessary to initialize a window.

- **glutInit**(int **argc*, char ***argv*) initializes GLUT and processes any command line arguments (for X, this would be options like –display and -geometry). It should be called before any other GLUT routine.
- **glutInitDisplayMode**(*unsigned int mode*) specifies whether to use an RGBA or color-index color model. We can also specify whether we want a single or double buffered window. GLUT SINGLE: selects a single-buffered window which is the default if isn't called;GLUT DOUBLE: selects a double-buffered window.
- **glutInitWindowPosition**(int*x*,int*y*) specifies the screen location for the upper-left corner of your window.
- **glutInitWindowSize** (int*width*,int*size*) specifies the size, pixels, of window.
- **glutCreateWindow** (char **string*) creates a window with an OpenGL context.

5.5 CALLBACK FUNCTIONS

All callback function more often just called a callback, in C function, written by the application programmer. But there's one important difference between a callback function and an ordinary C function: application never calls the callback function directly. Instead, the callback function is called by OpenGL.

5.5.1 THE DISPLAY CALLBACK:

glutDisplayFunc (void (**func*)(void)) is the first and most important event callback function. Whenever GLUT determines the contains of the window to be redisplayed.

- The callback function registered by glutDisplayFunc () is executed. Therefore, we should put all the routines you need to redraw the scene in the display callback function.
- Void glutDisplayFunc(void(**func*)(void));

glutDisplayFunc () registers the name of the callback function to be invoked when OpenGL needs to redisplay the contents of the window. The application must register a display function – it is not optional.

If your program changes the contents of the window, sometimes you will have to call glutPostRedisplay(*void*), which gives glutMainLoop () a nudge to call the registered display callback at its next opportunity.

5.5.2 THE KEYBOARD CALLBACK:

- void **glutKeyboardFunc**(void(**func*) (unsigned char *key*, intx, inty));
- **glutKeyboardFunc ()** registers the application function to call when OpenGL detects a key press generating an ASCII character. This can only occur when the mouse focus is inside the OpenGL window. It expects a function func () which returns void, and has three arguments key, x and y. So, it's function is:

```
void keyboard (unsigned char key, intx, inty)
{
/* called when a key is pressed */
}
```

Three values are passed to the callback function: key is the ASCII code of the key pressed; x and y give the pixel position of the mouse at the time. Inside the keyboard() callback, we look at the value of key. If it's 27 we call the standard C function exit () to terminate the program cleanly.

5.5.3 THE MOUSE CALLBACK

- void **glutMouseFunc** (void (**func*) (int*button*, int*state*, int*x*, int*y*));

glutMouseFunc () register an application callback function which GLUT will call when the user presses a mouse button within the window. The following values are passed to the callback function:

- button records which button was pressed, and can be
  -GLUT LEFT BUTTON
  -GLUT MIDDLE BUTTON
  -GLUT RIGHT BUTTON
- state records whether the event was generated by pressing the button (GLUT DOWM), or releasing it (GLUT UP).
- x,y give the current mouse position in pixels. Note: when using OpenGL with X, the mouse y position is measured from the top of the window.

5.5.4 HANDLING INPUT EVENTS:

You can use these routines to register callback commands that are invoked when specified events occur.

- **glutReshapeFunc**(void(**func*) (int*w*, inth)) indicates what action should be taken when the window is resized.
- **glutKeyboardFunc**(void(**func*)(unsignedchar*key*,int*x*,int*y*))and **glutMouseFunc**(void(*func)(int*button*,int*state*,int*x*,int*s*)) allow you to link akeyboard key or a mouse button with routine that is invoked when the key or mouse button is pressed or released.
- **glutMotionFunc** (void (**func*) (int*x*,int*y*)) registers a routine to call back when the mouse is moved while a mouse button is also pressed.

5.5.5 TRANSLATION:

- void **glTranslatef**(GLfloat*x*,GLfloat*y*,GLfloat*z*);
  glTranslatef () creates a matrix M which performs a translation by (x,y,z) and then post-multiplies the current matrix by M.

5.5.6 SCALING:

- void **glScalef** (GLfloat*x*, GLfloaty, GLfloatz);
  glScalef() creates a matrix M which performs a scale by (x,y,z) and then post-multiplies the current matrix by M.

5.5.7 ROTATION:

- void **glRotatef**(GLfloat*angle*,GLfloat*x*,GLfloat*y*,GLfloat*z*);
  glRotate() creates a matrix M which performs a counter-clockwise rotation of angle in degrees. The axis about which the rotation occurs in the vector from the origin (0, 0, 0) to the point (x,y,z),and then post-multiplies the current matrix by M.

5.5.8 PUSH AND POP OPERATIONS:

There are two functions which operate on the current matrix stack:

glPush Matrix()

glPopM atrix()

void **glPushMatrix(*void*);** Pushes the current matrix stack down one level. The matrix on the top of the stack is copied into the next-to-top position.

void **glPopMatrix(*void*);** Pops the current matrix stack, moving each matrix in the stack one position towards the top of the stack.

The current matrix stack is determined by the most recent call to **glMatrixMode().**

5.6 RUNNING THE PROGRAM

After all the setup is completed, GLUT programs enter an event processing loop,

glutMainLoop().

void **glutMainLoop**(void) enters the GLUT processing loop, never to return. Registered callback functions will be called when the corresponding events instigate them.

CHAPTER 6

IMPLEMENTATION

```
#include <stdio.h>
#include <math.h>
#include <GL/glut.h>
#define pi 3.142
static GLfloat angle = 0;
static int submenu;
static int mainmenu;
static int value = -1;
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)
{
 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)
{
 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)
{
 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);
 glColor3f(0, 0, 1);
```

```
 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);
 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);
 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);
 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);
 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);
 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);
 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);
 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);
 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, 0.1, 0.9);
 if (value == -1)
 {
 char cn[] = "DON BOSCO INSTITUTE OF TECHNOLOGY ";
 drawhead(-490, 900, 0, cn);
 char pn[] = "Kumbalagodu, Bangalore- 560060";
 drawsubhead(-250, 850, 0, pn);
 char dn[] = "DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING";
 drawhead(-690, 650, 0, dn);
 char prn[] = "A Mini Project On";
 drawsubhead(-150, 450, 0, prn);
 char pro[] = "ATOM SIMULATION";
 drawhead(-250, 350, 0, pro);
 char pb[] = "PROJECT BY: ";
 drawhead(-690, -150, 0, pb);
 char p1[] = "Latharani";
 drawhead(-690, -250, 0, p1);
 char p1u[] = "1db20CS405";
 drawsubhead(-690, -300, 0, p1u);
 char gb[] = "GUIDED BY: ";
 drawhead(290, -150, 0, gb);
 char g1[] = "Mr. Vishesh";
 drawhead(290, -250, 0, g1);
 char d1[] = "Associate Professor, Dept. Of CSE, DBIT";
 drawsubhead(290, -300, 0, d1);
 char in[] = "Press enter to Continue";
 drawhead(-250, -700, 0, in);
 glutSwapBuffers();
 glutDetachMenu(GLUT_RIGHT_BUTTON);
```

```
}
if (value != -1)
{
nuc(250);
char n[] = "NUCLEUS";
drawString(-90, 20, 0, n);
char d[] = "(NEUTRON + PROTON)";
drawString(-225, -30, 0, d);
if (value == 0)
{
char nu[] = "SELECT THE ELEMENT USING MENU";
drawhead(-490, 900, 0, nu);
}
}
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();
}}
```

CHAPTER 7
RESULTS AND SNAPSHOTS

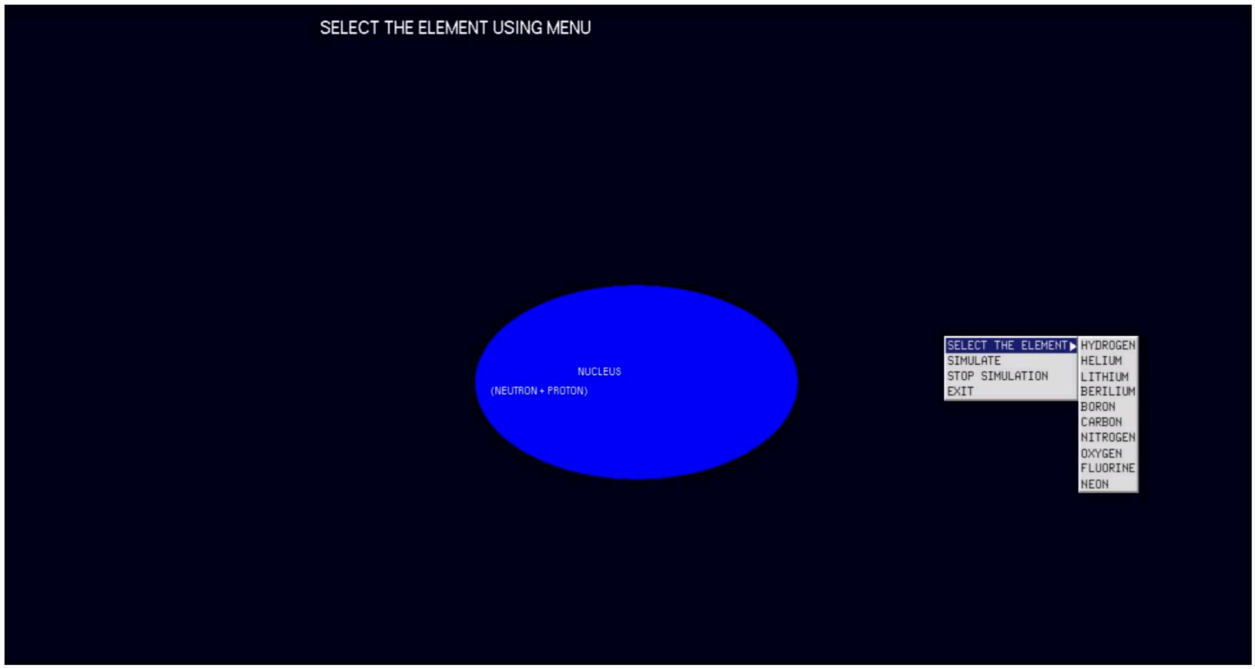


Fig 7.1 Atom simulation

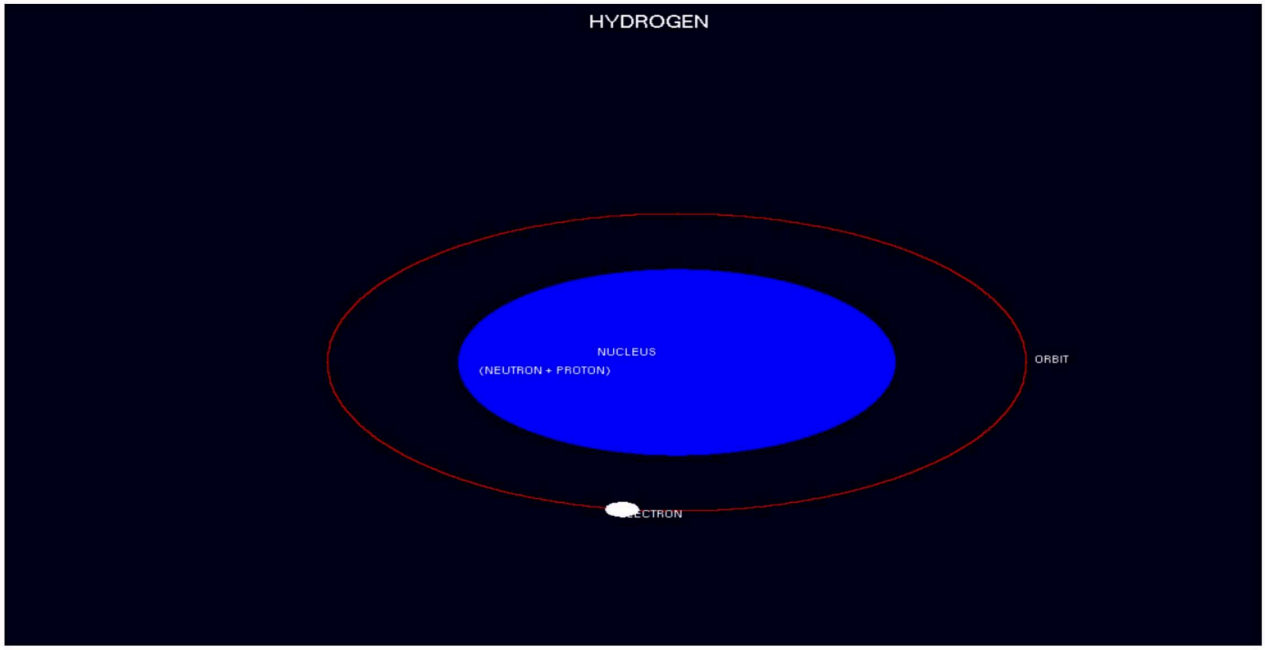


Fig 7.2 Hydrogen Simulation

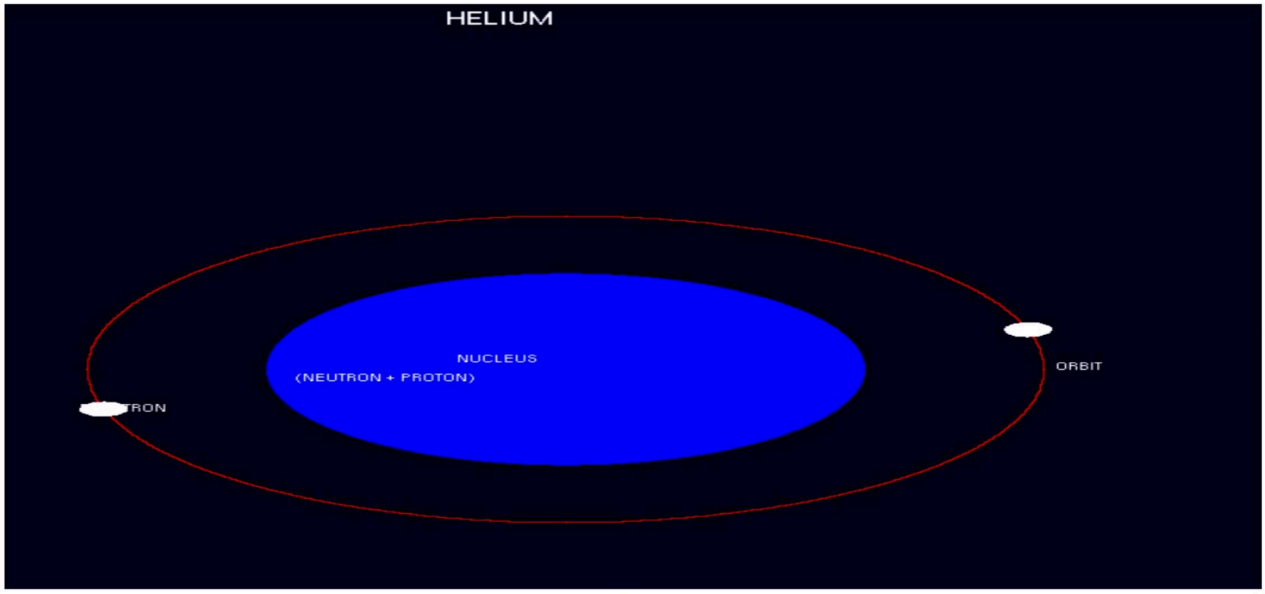


Fig 7.3 Helium Simulation

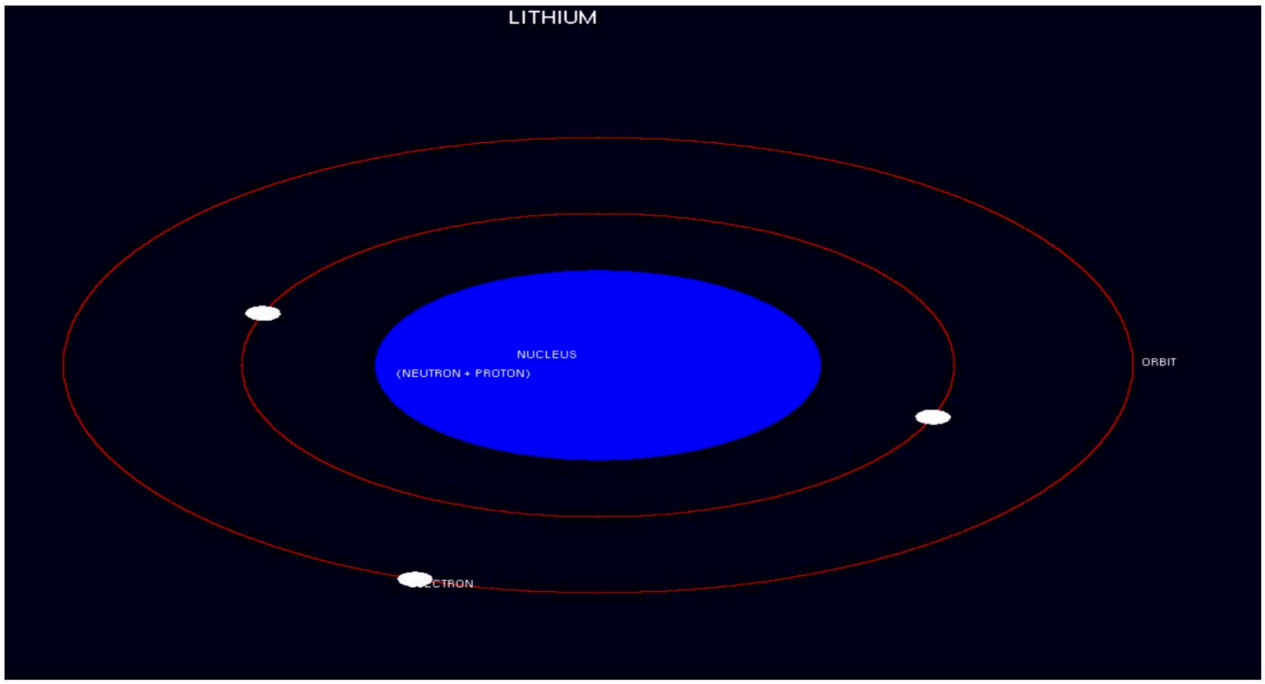


Fig 7.4 Lithium Simulation

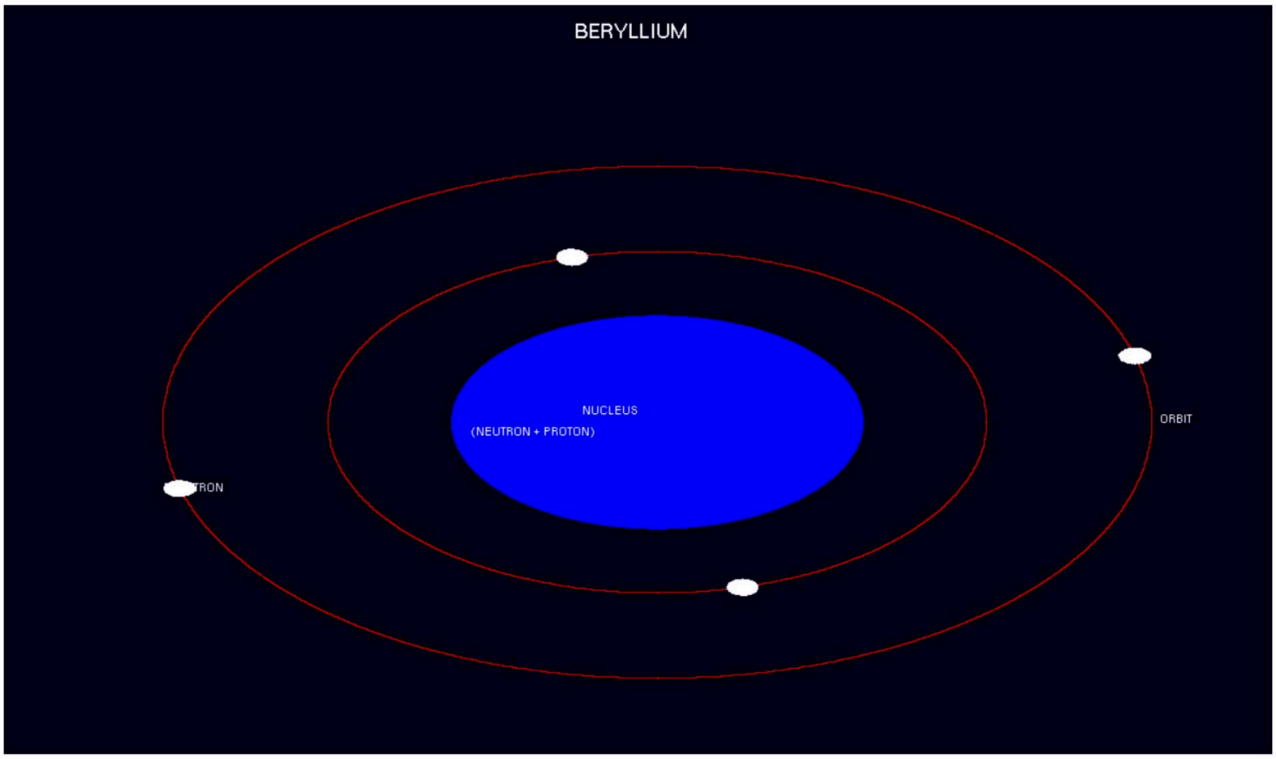


Fig 7.5 Beryllium Simulation

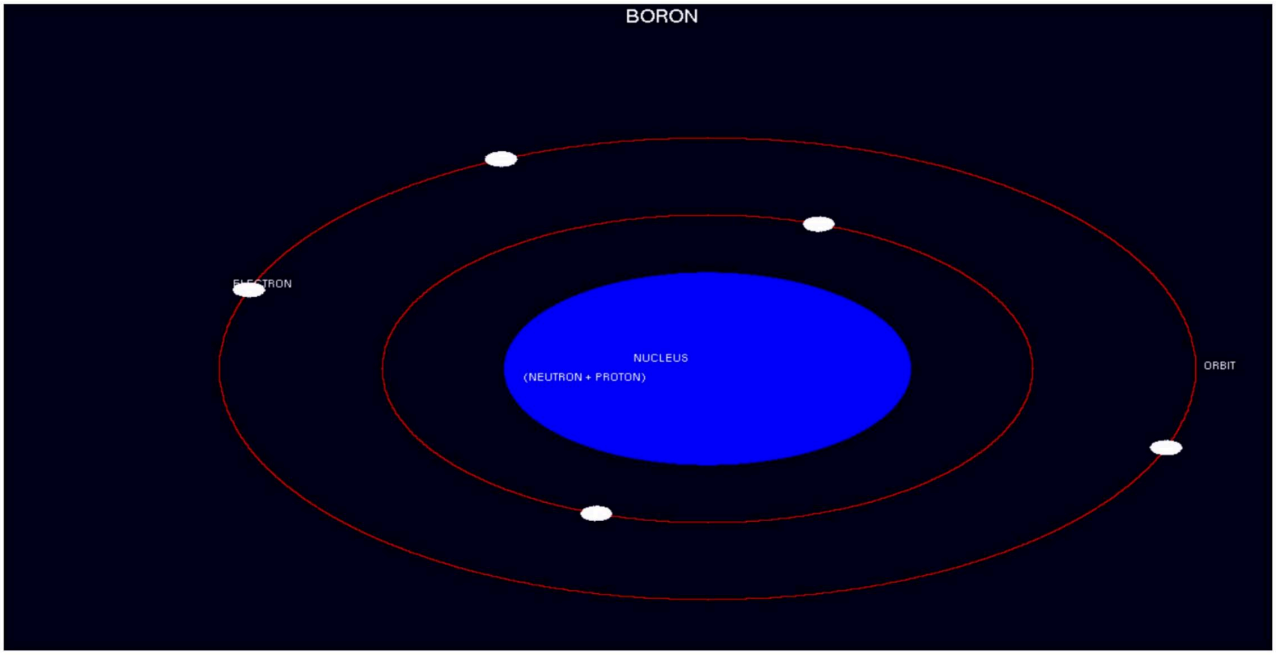


Fig 7.6 Boron Simulation

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 also, 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.

The number of electrons in valence shell which can be lost or shared or accepted can be called as valency of that element.

Also the number of electrons required by the valence shell to complete duplet or octet is called as the valency

REFERENCES

Books:

Edward Angel, "Interactive Computer Graphics",5th edition, Pearson Education,2005

Donald D Hearn and **M. Pauline Baker,** "Computer Graphics with OpenGL", 3rd Edition

Websites:

- **http://www.opengl.org**
- **http://glprogramming.com/red**
- **http://jerome.jouvie.free.fr/OpenGl**

https://www.3dgep.com/introduction-to-opengl-and-glsl/