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 presentation.

Computer-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 Aim

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.

1.4 Introduction to OpenGL

OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same

set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

Computer graphics, a 3-dimensional primitive can be anything from a single point to an n- sided polygon. From the software standpoint, primitives utilize the basic 3dimensional rasterization algorithms such as Bresenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth. OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives. OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

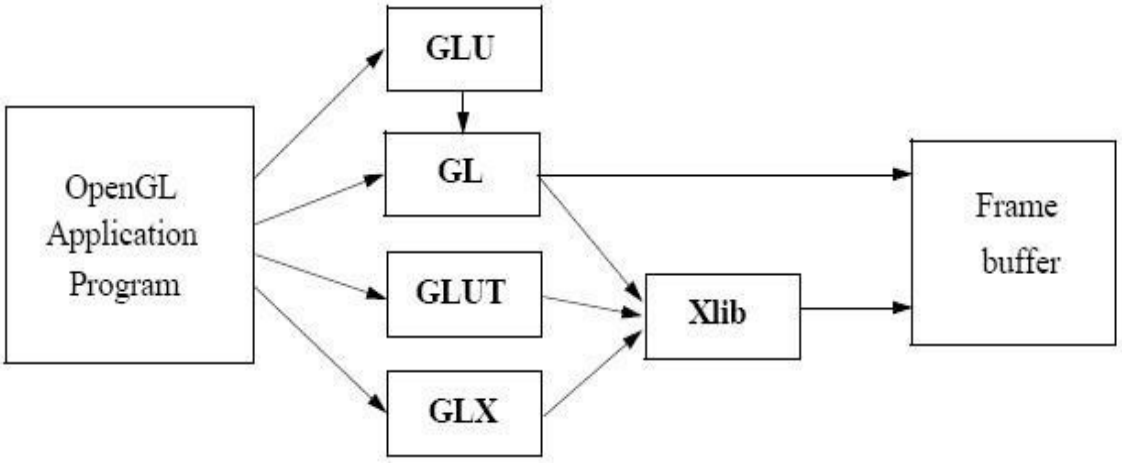

Figure 1.1: Basic block diagram of OpenGL

GLUT

GLUT, short for OpenGL Utility Toolkit, is a set of support libraries available on every major platform. OpenGL does not directly support any form of windowing, menus, or input. That's where GLUT comes in. It provides basic functionality in all of those areas, while remaining platform independent, so that you can easily move GLUT-based applications from, for example, Windows to UNIX with few, if any, changes.

1.5 Project Related Concepts

The objective is to build an atom simulator which can convince the audience about the structure of an element. The coding is implemented for the atoms from Hydrogen to Neon, that is for 10

elements. In this simulation importance is given on a structure of an element and how the electrons revolve around the nucleus.

The basic requirements of the atom simulator are analyzed to be:

- **User Interface**- User should be able to select an element and start the simulation on their own. They can start, stop and change the elements of their choice and after this they can exit the simulation.
- **Element selection**- User can select the element from Hydrogen to Neon for the simulation, that is from atomic number 1 to 10.
- **Start/Stop Simulation**- User after selecting an element from the mentioned list he/she can start the simulation. As soon as they select start, the electrons around the nucleus starts revolving around the nucleus within their orbit. If they select the stop simulation option, the simulation will be stopped.

1.6 Interfaces

Mouse Interface

Select the element: The user clicks the right click button on the mouse, the screen will be prompted with the list of options. First option is to select the user desired element from the list. The list contains elements from Hydrogen to Neon for the simulation, that is from atomic number 1 to atomic number 10.

Simulate: After user selecting an element, when he/she clicks on the simulate option, the electrons around the nucleus starts revolving around the nucleus within their orbit.

Stop simulation: If a user selects this option, the simulation will be paused.

Exit: The program execution will be terminated and the window will be destroyed after selecting this option.

Keyboard Interface: Three functionalities are implemented using the keyboard function.

- After selecting an element, if a user presses spacebar the simulation will be started.
- After starting the simulation if the user clicks on ‘S’ key, simulation will be paused.
- If the user clicks on the ‘Q’ key, program execution will be terminated and the window will be destroyed.

CHAPTER 2

REQUIREMENTS SPECIFICATION

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.

2.1 Software Requirements

- An MS-DOS based operating system like Windows 98, Windows 2000 or WindowsXP, vista, windows 7 is the platform required to develop the 2D and 3D graphics applications.
- A Visual C/C++ compiler is required for compiling the source code to make the executable file which can then be directly executed.
- A built in graphics library like glut and glut32, and header file like GL\glut.h and also dynamic link libraries like glut and glut32 are required for creating the 3D layout.

2.2 Hardware Requirements

The hardware requirements are very minimal and software can run on most of the machines

- Processor Speed – 500MHz or above
- RAM – 64MB or above Storage Space – 2 MB or above, hard disk – 10MB
- Monitor resolution – A color monitor with a minimum resolution of 1000*700
- Support both single and double buffering.

CHAPTER 3

DESIGN

3.1 Window design

Atom simulation uses only one window. That is Atom simulation (Main window): This window contains all the contents that is menu bar and simulation display. This is window used for all the events and functions in this project. In this window we display simulation of first 10 atoms in the periodic table. And all mouse and keyboard events triggered in this window. All the labels and Information about the model will be displayed on this window.

3.2 Menu bar

Menu bar is designed so that one can easily access the various options like selecting the elements or starting the simulation or stopping the simulation etc.

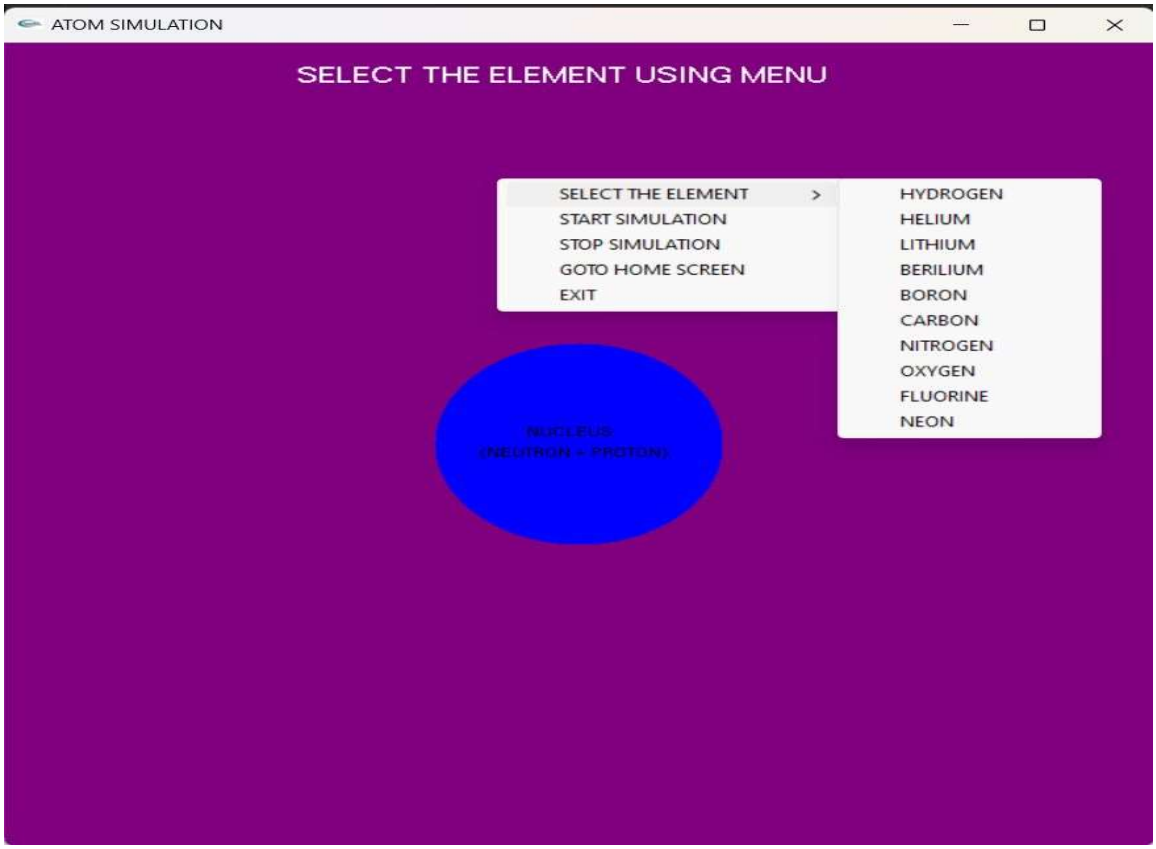

Figure 3.1: Menu Bar

3.3 Simulation display

As soon as user selects the element and click on the simulate option, the electrons around the nucleus starts revolving around the nucleus within their orbit.

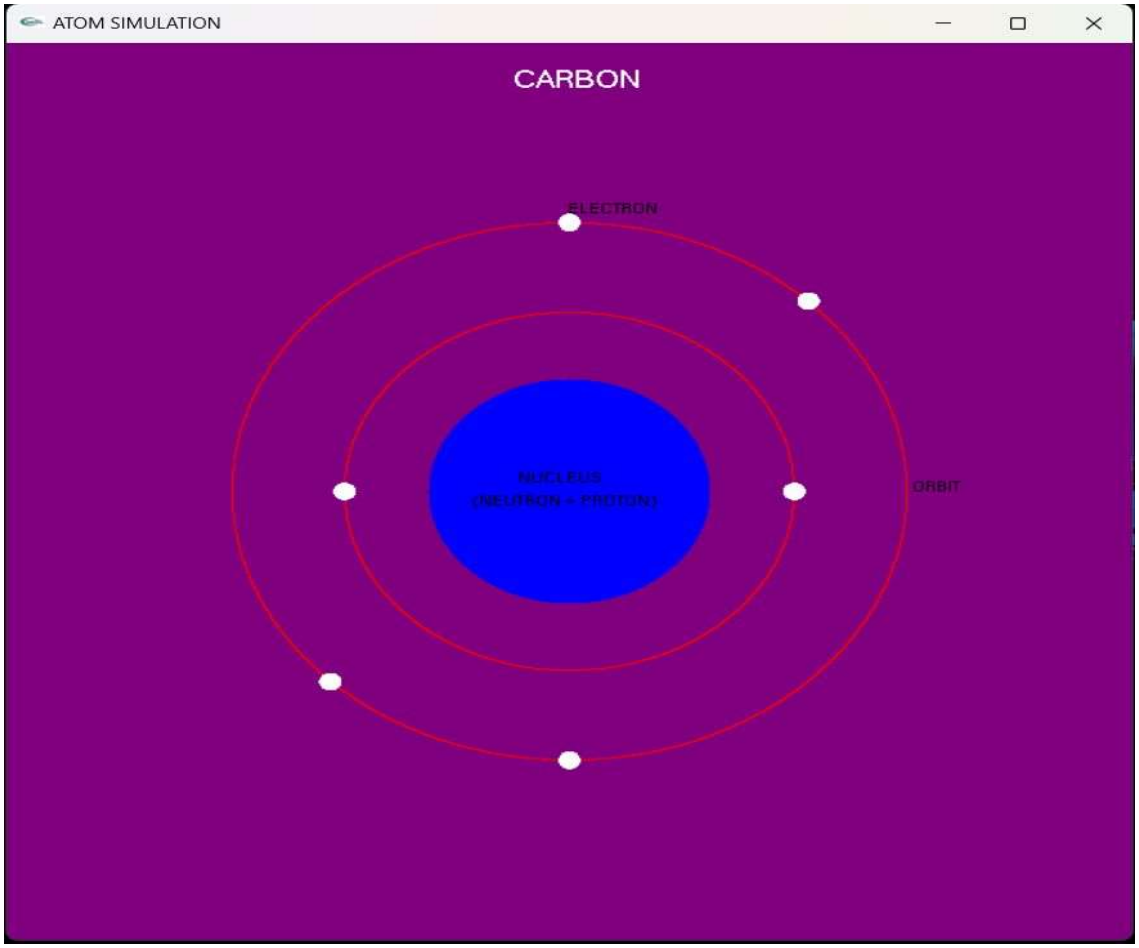

Figure 3.2: Simulation Display

CHAPTER 4

IMPLEMENTATION

4.1 Functions used glRasterPos3f(x, y, z):

OpenGL maintains a 3-D position in window coordinates. This position, called the raster position, is maintained with subpixel accuracy. The current raster position consists of three window coordinates (x, y, z), a clip coordinate w value, an eye coordinate distance,a valid bit, and associated color data and texture coordinates.

glutAddMenuEntry(args): glutAddMenuEntry adds a menu entry to the bottom of the current menu. The string name will be displayed for the newly added menu entry. If the menu entry is selected by the user, the menu's callback will be called passing value as the callback's parameter.

glutAttachMenu(button): glutAttachMenu attaches a mouse button for the current window to the identifier of the current menu; glutDetachMenu detaches an attached mouse button from the current window. By attaching a menu identifier to a button, the named menu will be popped up when the user presses the specified button. Button should be one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON and GLUT_RIGHT_BUTTON. Note that the menu is attached to the button by identifier, not by reference.

glutMouseFunc(args): glutMouseFunc sets the mouse callback for the current window. When a user presses and releases mouse buttons in the window, each press and each release generates a mouse callback. The button parameter is one of GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON, or GLUT_RIGHT_BUTTON. For systems with only two mouse buttons, it may not be possible to generate GLUT_MIDDLE_BUTTON callback. For systems with a single mouse button, it may be possible to generate only a GLUT_LEFT_BUTTON callback. The state parameter is either GLUT_UP or GLUT_DOWN indicating whether the callback was due to a release or press respectively. The x and y callback parameters indicates the window relative coordinates when the mouse button state changed. If a GLUT_DOWN callback for a specific button is triggered, the program can assume a GLUT_UP callback for the same button will be generated (assuming the window still has a mouse callback registered) when the mouse button is released even if the mouse has moved outside the window.

glutKeyboardFunc(args): 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. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks. During a keyboard callback, glutGetModifiers may be called to determine the state of modifier keys when the keystroke generating the callback occurred. callback was due to a release or press respectively. The x and y callback parameters indicates the window relative coordinates when the mouse button state changed. If a GLUT_DOWN callback for a specific button is triggered, the program can assume a GLUT_UP callback for the same button will be generated (assuming the window still has a mouse callback registered) when the mouse button is released even if the mouse has moved outside the window.

glutKeyboardFunc(args): 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. The state of modifier keys such as Shift cannot be determined directly; their only effect will be on the returned ASCII data. The x and y callback parameters indicate the mouse location in window relative coordinates when the key was pressed. When a new window is created, no keyboard callback is initially registered, and ASCII key strokes in the window are ignored. Passing NULL to glutKeyboardFunc disables the generation of keyboard callbacks. During a keyboard callback, glutGetModifiers may be called to determine the state of modifier keys when the keystroke generating the callback occurred

CHAPTER 5

TESTING

Test Case ID	Steps to execute the Test Case	Expected Result	Actual Result	Remarks
1.	Mouse right click	It shows menu bar to user.	It shows menu bar to user.	Pass
2.	Selecting the options 1.Select element 2.Start simulation 3.Stop simulation 4.Exit 5.Go to Home	It shows the list of option to user from which They can select an option. Selects the element in the given list. Starts the simulation. Stops the simulation. Exits from the window. Display starting window	It shows the list of option to user from which They can select an option. Selects the element in the given list. Starts the simulation. Stops the simulation. Exits from the window. Display starting window	Pass

Table 5.1 Test cases for Mouse interface

Test Case ID	Steps to execute the Test Case	Expected result	Actual Result	Remarks
1.	Choosing the Option 1.Simulate 2.Stop simulation 3. Exit	It shows the list of option to user from which they can select an option. Starts the simulation. (Space bar) Stops the simulation. (‘S’) Exits from the window(‘Q’)	It shows the list of option to user from which they can select an option. Starts the simulation. (Space bar) Stops the simulation. (‘S’) Exits from the window(‘Q’)	Pass

Table 5.2 Test cases for Keyboard interface

CHAPTER 6

SNAPSHOTS

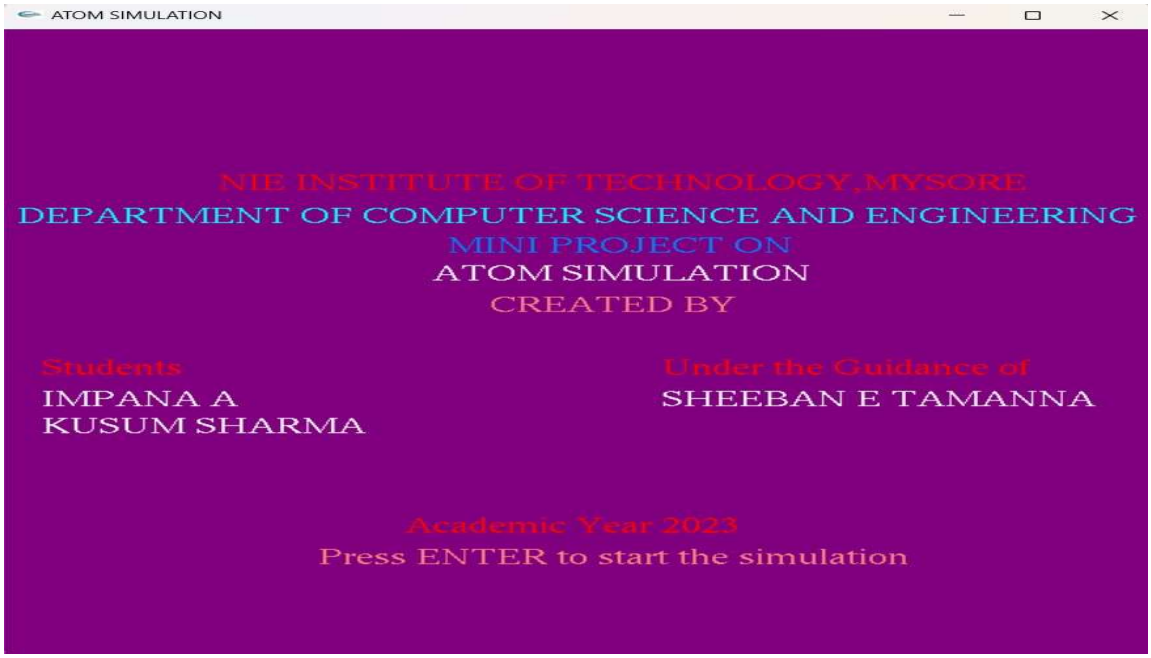

Figure 6.1: Home Screen

Click Enter to move to the next page.

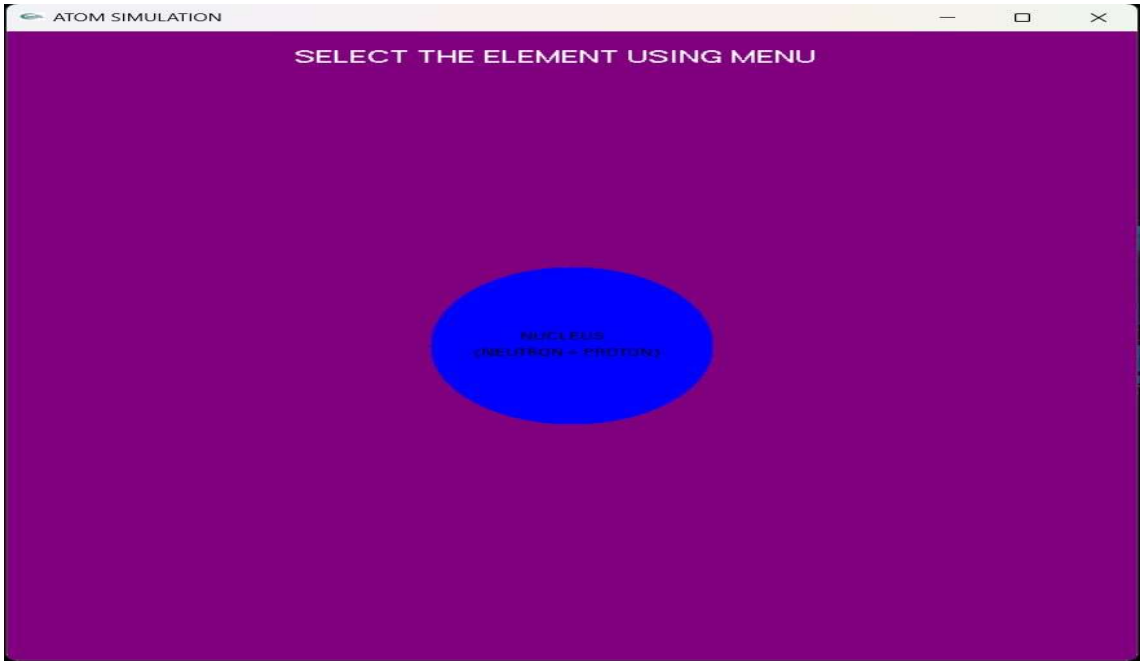

Figure 6.2: Starting Screen

Starting screen displays the nucleus orientation.

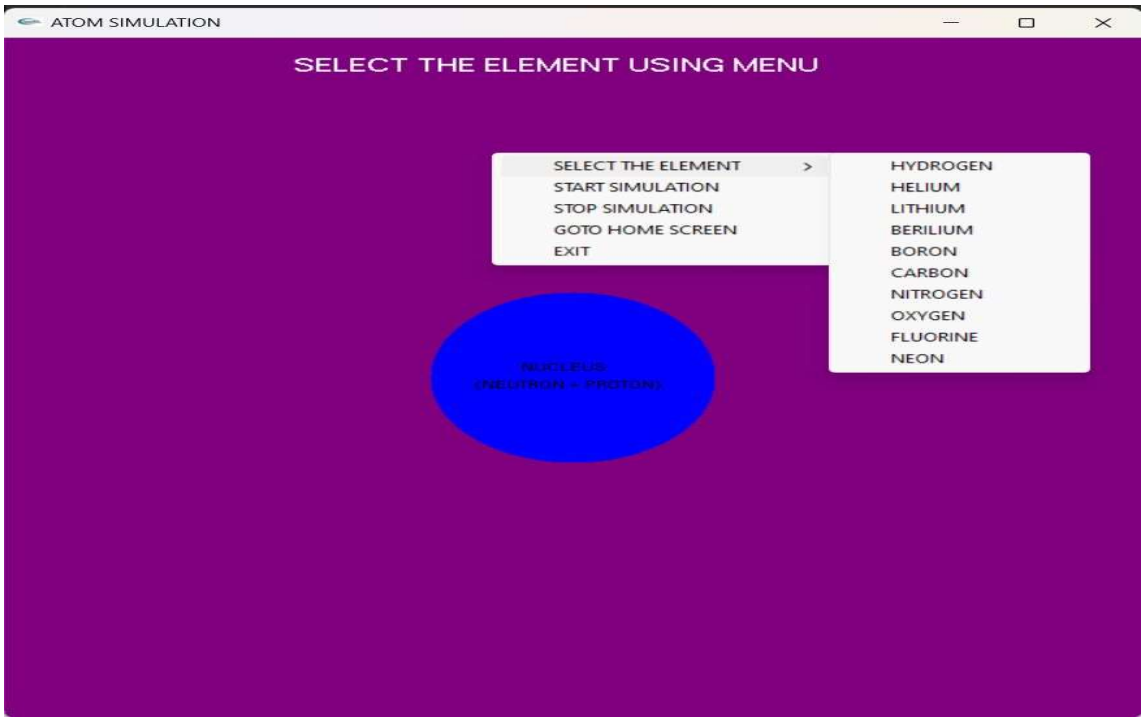

Figure 6.3: Menu Interface

Menu interface contains the options required for the simulation like start, stop and Exit.

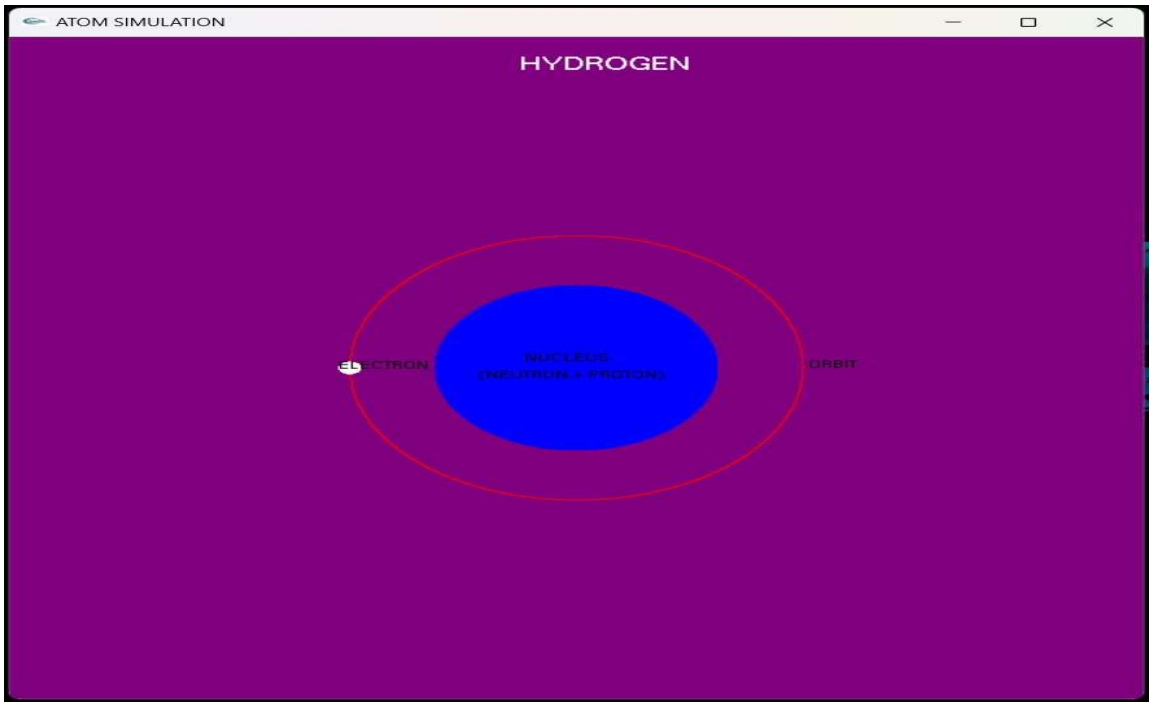

Figure 6.4: Hydrogen Simulation

Hydrogen contains 1 electron and revolving of this electron around the nucleus is shown here.

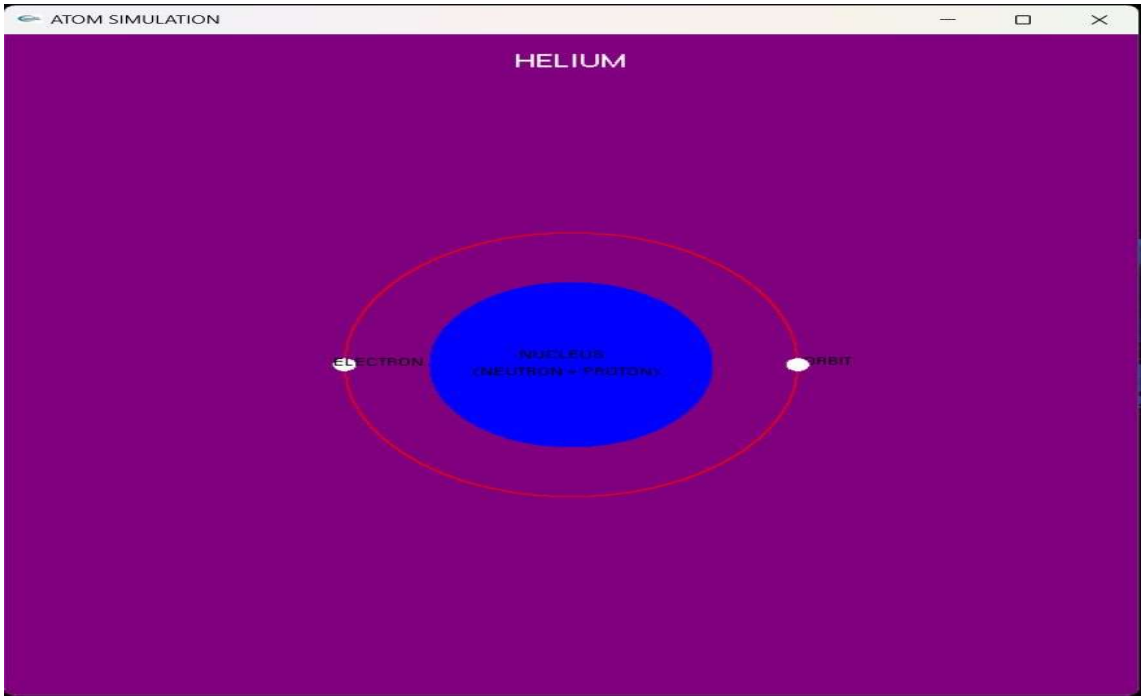

Figure 6.5: Helium Simulation

Helium contains 2 electrons and revolving of these electrons around the nucleus is shown here.

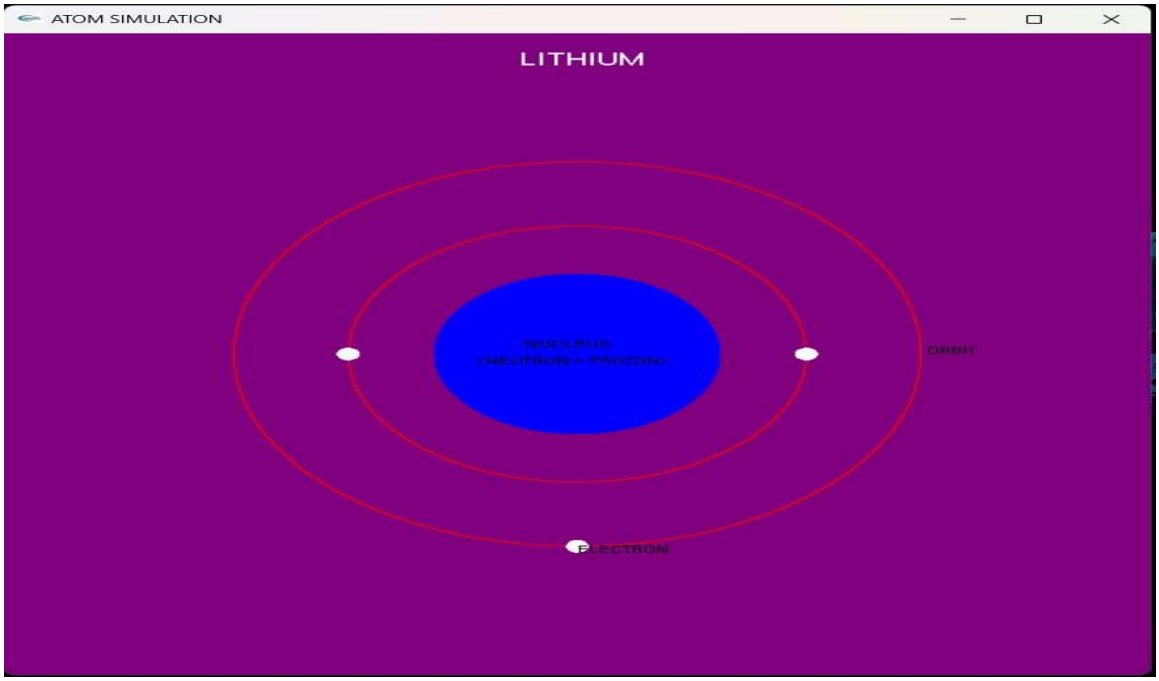

Figure 6.6: Lithium Simulation

Lithium contains 3 electrons and revolving of these electrons around the nucleus is shown here.

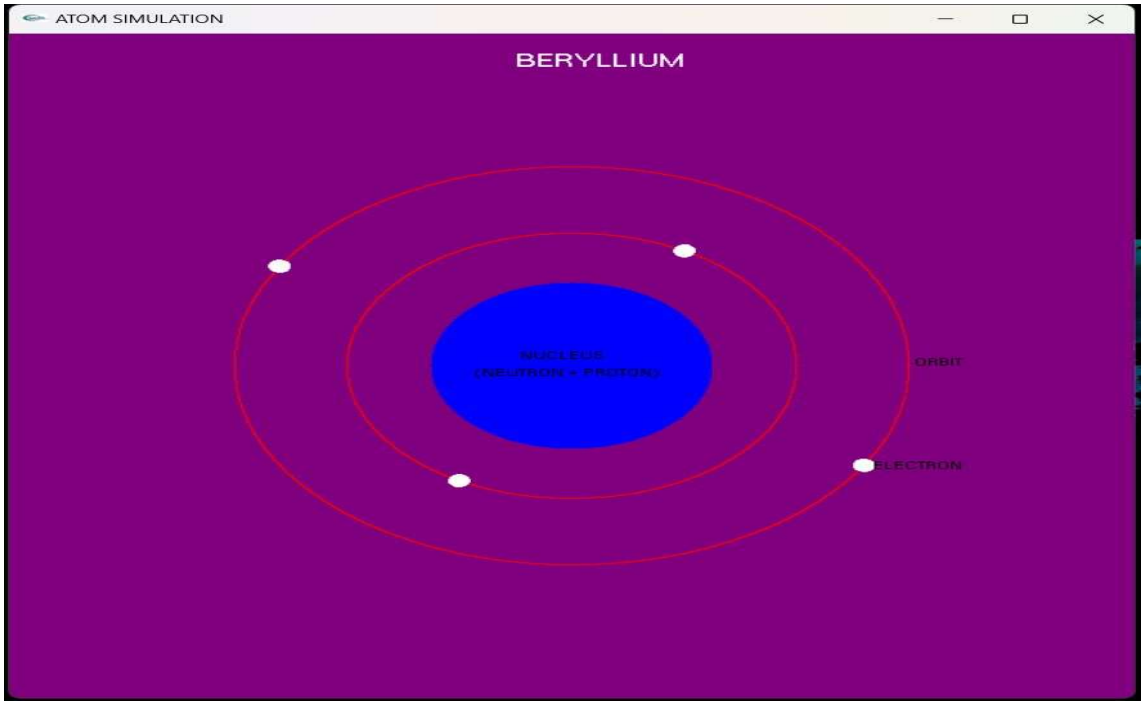

Figure 6.7: Beryllium Simulation

Beryllium contains 4 electrons and revolving of these electrons around the nucleus is shown here.

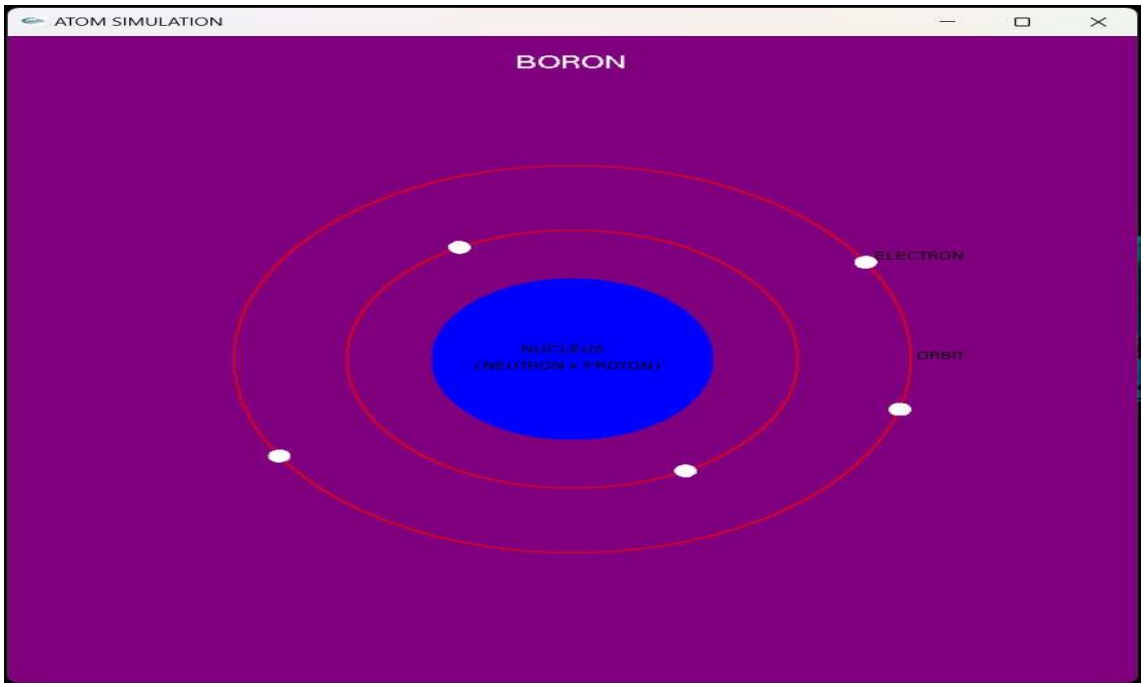

Figure 6.8: Boron Simulation

Boron contains 5 electrons and the revolving of these electrons around the nucleus is shown here.

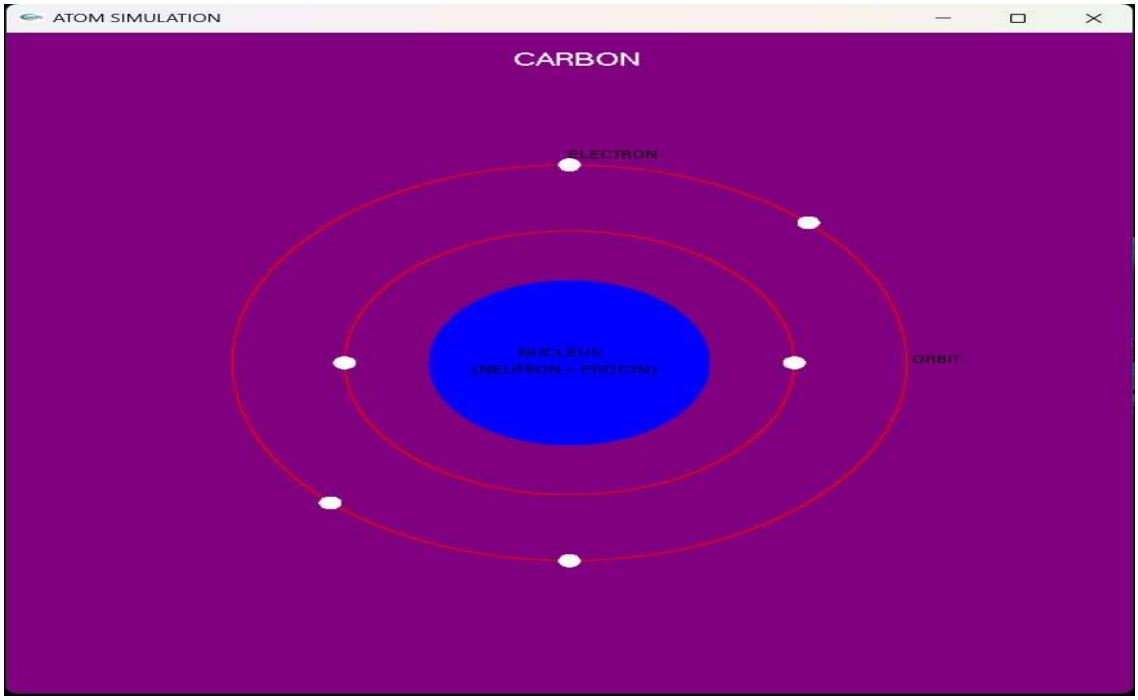

Figure 6.9: Carbon Simulation

Carbon contains 6 electrons and revolving of these electrons around the nucleus is shown here.

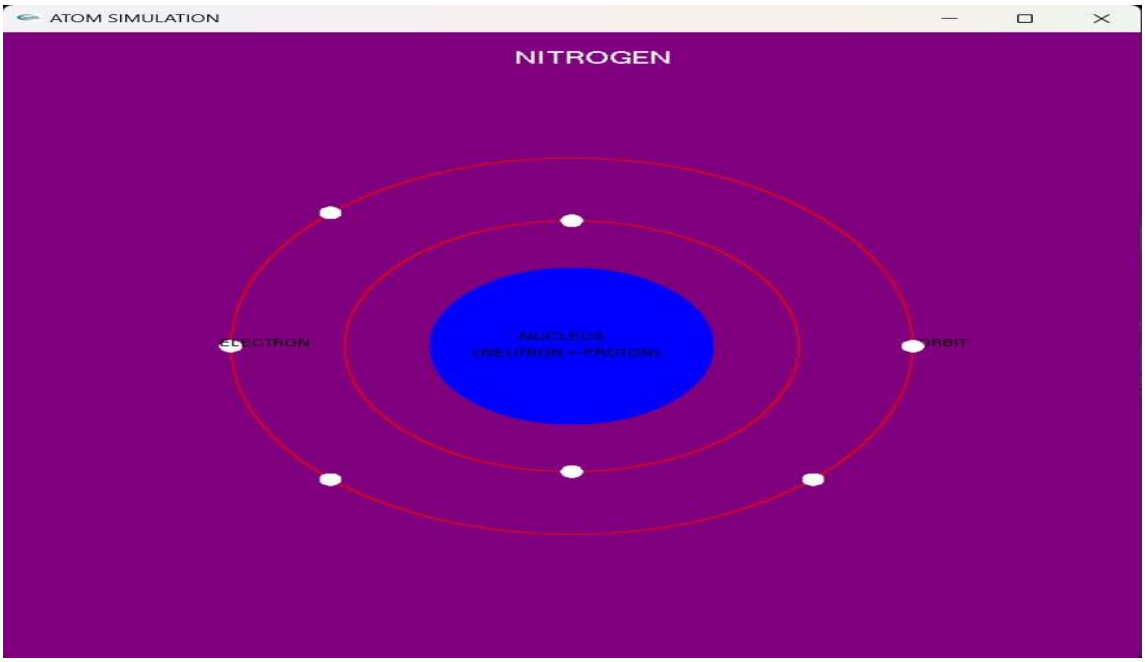

Figure 6.10: Nitrogen Simulation

Nitrogen contains 7 electrons and revolving of these electrons around the nucleus is shown here.

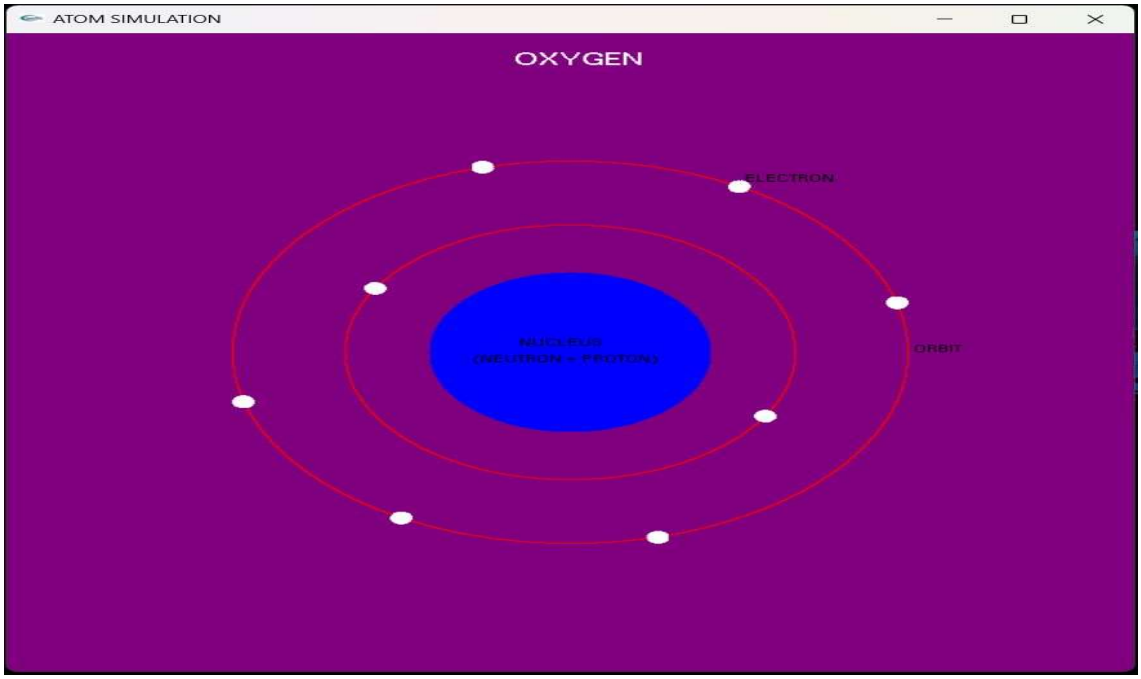

Figure 6.12: Oxygen Simulation

Oxygen contains 8 electrons and revolving of these electrons around the nucleus is shown here.

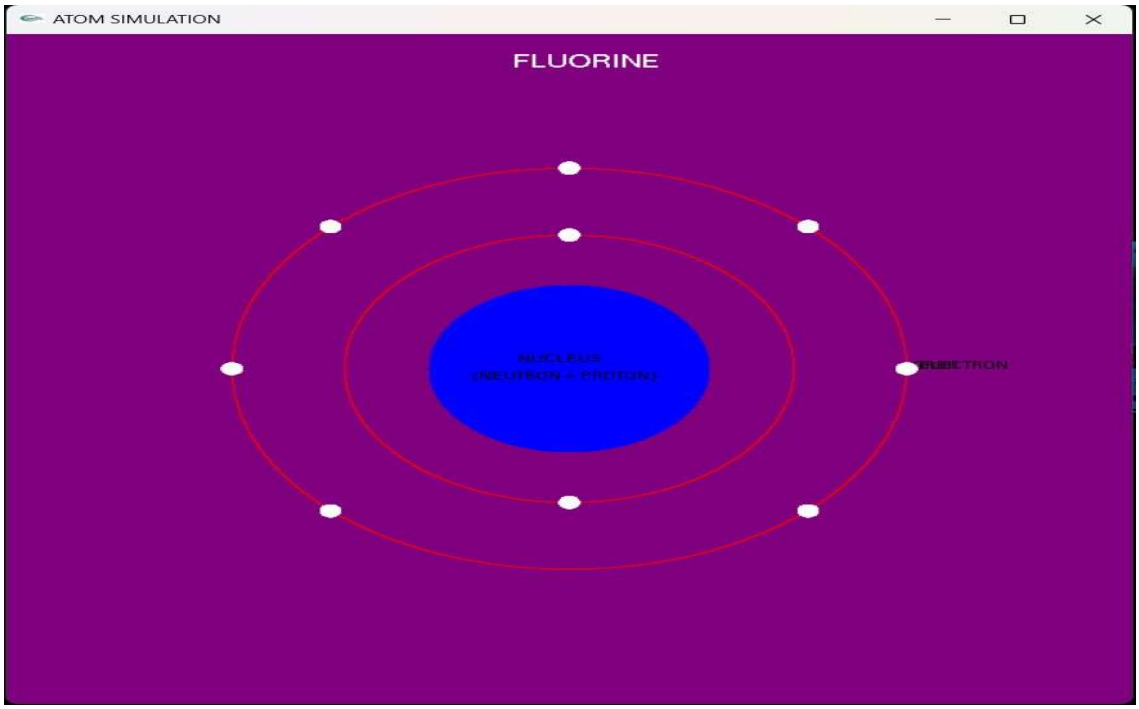

Figure 6.12: Fluorine Simulation

Fluorine contains 9 electrons and revolving of these electrons around the nucleus is shown here.

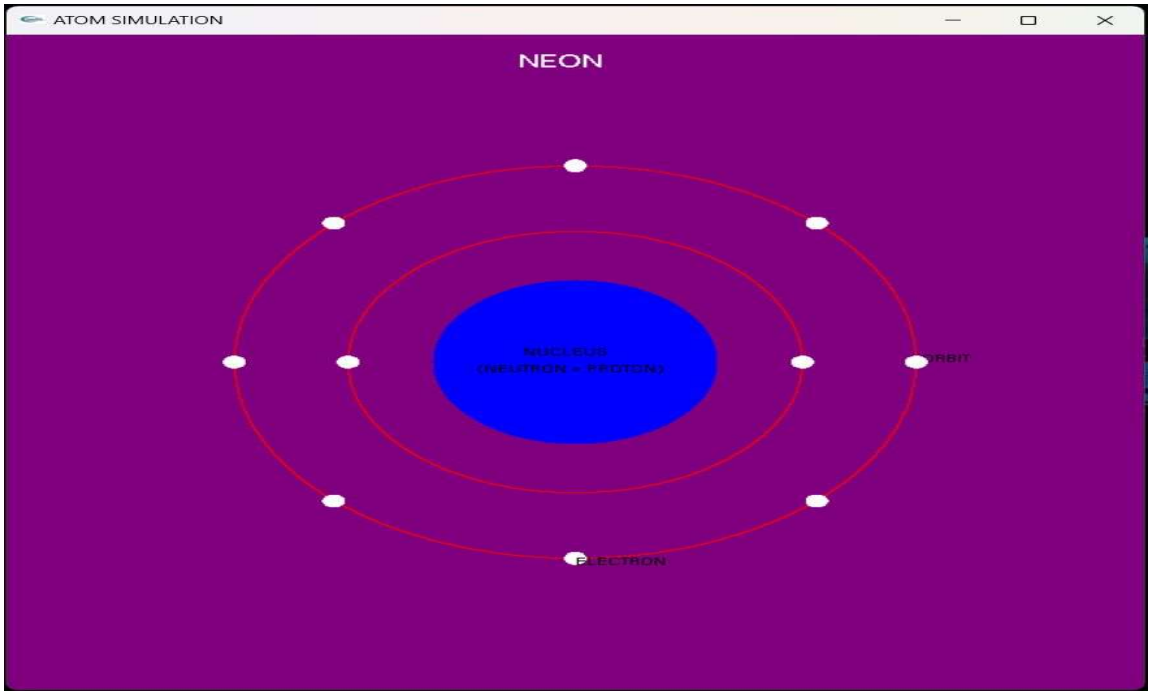

Figure 6.13: Neon Simulation

Neon contains 10 electrons and revolving of these electrons around the nucleus is shown here.

CHAPTER 7

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.

FUTURE ENHANCEMENTS

We can add many features in the atom simulation project which will be supported in the future versions which includes adding all the elements from the periodic table .presently there are 10 elements, also we can show the simulation with all the important details of an element, adding a search bar for searching the elements from the list of all the elements. We can make 3-D simulation to make the project much more interactive, other features includes forming a compound element by combining 2 or more simple elements also we can include the attraction and repulsion of 2 elements with opposite or same charge.

BIBLIOGRAPHY

- Interactive Computer Graphics A Top-Down Approach with OpenGL -Edward Angel, 5th Edition, Addison-Wesley, 2008.
- Computer Graphics Using OpenGL – F.S. Hill Jr. 2nd Edition, Pearson Education, 2001.
- Computer Graphics – James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, Addison-Wesley 1997.
- Computer Graphics - OpenGL Version – Donald Hearn and Pauline Baker, 2nd Edition, Pearson Education, 2003.
- Fundamentals of Computer graphics- Steve Marschner, Peter Shirley, 4th Edition.

FURTHER LINKS:

- www.google.com
- www.openglforum.org