MIT | Academy of Engineering
(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

A Project Report on

ATOM SIMULATION

Submitted by,

SAKSHI BHINGARKAR	(PRN No. 202301040260)
VAISHNAVI THORAVE	(PRN No. 202301040261)
SAMADHAN MANE	(PRN No. 202301040262)

Guided by,

Mrs. Chetana Nemade

School of Computer Engineering and Technology

MIT Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

Alandi (D), Pune – 412105

(2023–2024)

Index

Introduction

1.This program uses OpenGL and GLUT to simulate atoms, including a nucleus and revolving electrons.
2.It creates an interactive window with a right-click menu to choose elements (Hydrogen, Helium, etc.).
3.The display function renders text and atom structure based on the selected element.
4.Keyboard and mouse controls allow for interactions such as rotating electrons, starting/stopping simulations, and changing window states.
5.The simulation is implemented using 2D circular orbits with color customization. It can be expanded or modified for additional visual effects or scientific representations.

Problem statement Design and implement a computer graphics program to simulate an **Atom** scene using OpenGL.

Aim

The aim of the code is to create an animated scene where:

-The Electron rotates .

-The Electron rotates stop on mouse command.

-The background elements remain static.

Operations:

- nuc(): Draws the nucleus using a colored polygon.
- circle(): Represents electron orbits as circles around the nucleus.
- eleright(), eleleft(), eletop(), eledown(), eletr(), eledl(), eledr(), eletl(): Functions to draw electrons in various positions around the nucleus for each element.
- Keyboard events (keyboard()): Enter starts the simulation, s stops rotation, q exits, and space resumes rotation.
- Mouse events (mouseControl()): The left mouse button initiates rotation.
- **Additional Notes:** -OpenGL commands are used to draw various shapes and objects.
- rotate(): Continuously rotates electrons around the nucleus, simulating the motion of electrons. The angle of rotation updates incrementally to create a smooth animation.
- Idle Functions: Used to start and stop animation based on user input from the keyboard and menu.

-The main function sets up the window, initializes the display, and starts the main loop to manage events

Objects:

- Nucleus: Represented as a central polygon (in pinkish-red color) in the atom structure. This nucleus is static and displayed at the center, symbolizing the protons and neutrons in an atom.
- Electrons: Represented as small circles that revolve around the nucleus. There are different functions (eleright, eleleft, eletop, eledown, etc.) to position electrons on various orbit paths. The rotation of electrons around the nucleus creates an atomic simulation effect.
- Orbits: Circular paths (using the circle function) that indicate electron shells. Different elements (Hydrogen, Helium, etc.) have different numbers of orbits depending on their atomic structure.
- Text Labels: Text for displaying element names, orbit labels, and other messages. This is done using functions like drawString, drawhead, and drawsubhead to place text on the screen for the project title, element names, and instructional prompts.
- Menu System: Includes a right-click context menu created with createMenu() and menu(), allowing the user to select elements, start/stop the animation, go to the home screen, or exit. The menu dynamically updates the simulation according to user input.
- Angle of Rotation: The angle variable controls the rotation of electrons around the nucleus, simulating movement. This is updated in the rotate function, which is triggered by user interaction or key press.

Hardware And Software Required:-

•Hardware:

- CPU: Multi-core (Intel i5 or equivalent)
- RAM: 8 GB+
- GPU: Dedicated with OpenGL support
- Storage: 2 GB free

•Software:

- OS: Windows, macOS, or Linux
- OpenGL: Version 3.3+
- Compiler: GCC or MSVC
- IDE: Visual Studio, Code::Blocks, or Eclipse
- Libraries: GLUT/GLFW, GLEW, GLM

Code

```
#include<windows.h>
#include <iostream>
#include <math.h>
#include <GL/glut.h>
#include <dos.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)
{ //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(0, 0, 0);
  //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(1, 1, 0.9, 1);
     glClear(GL_COLOR_BUFFER_BIT);

    if (value == -1)
    {
        char cn[] = "MIT ACADEMY OF ENGINEERING";
        drawhead(-650, 900, 0, cn);
        char pn[] = "ALANDI ,PUNE -412105";
        drawsubhead(-250, 850, 0, pn);

        char dn[] = "DEPARTMENT OF COMPUTER 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);
```

```
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();
  }
```

```
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);
```

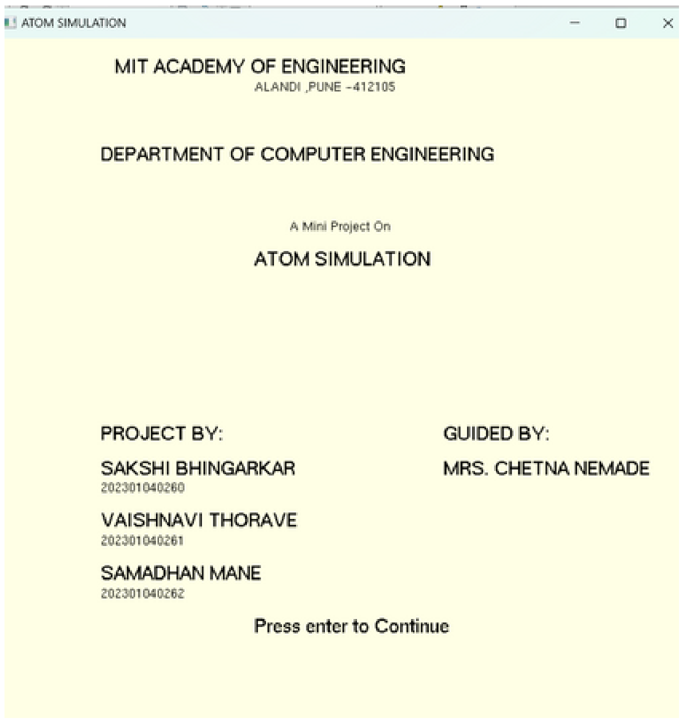
```
void fkey(int key, int x, int y)
{

    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);
```

```
 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;
}
```

Output



ATOM SIMULATION

MIT ACADEMY OF ENGINEERING
ALANDI ,PUNE -412105

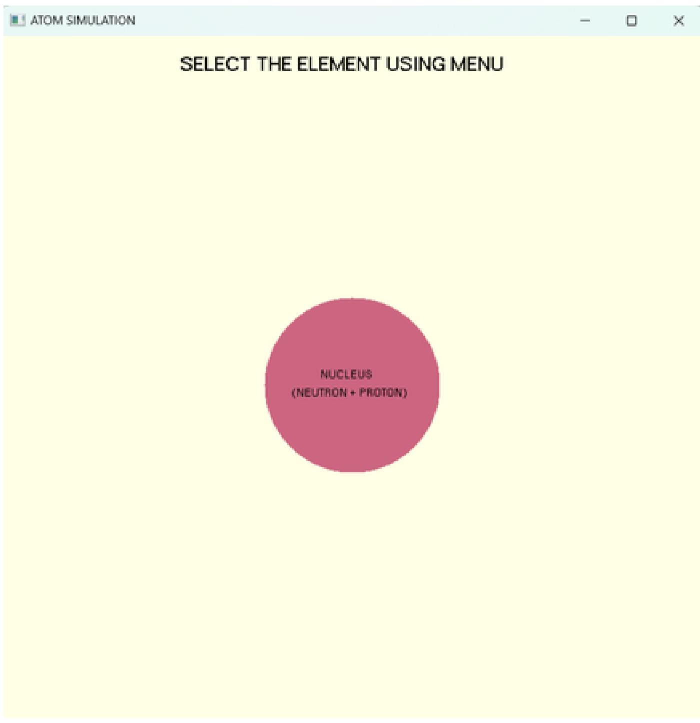
DEPARTMENT OF COMPUTER ENGINEERING

A Mini Project On
ATOM SIMULATION

PROJECT BY:
SAKSHI BHINGARKAR
202301040260
VAISHNAVI THORAVE
202301040261
SAMADHAN MANE
202301040262

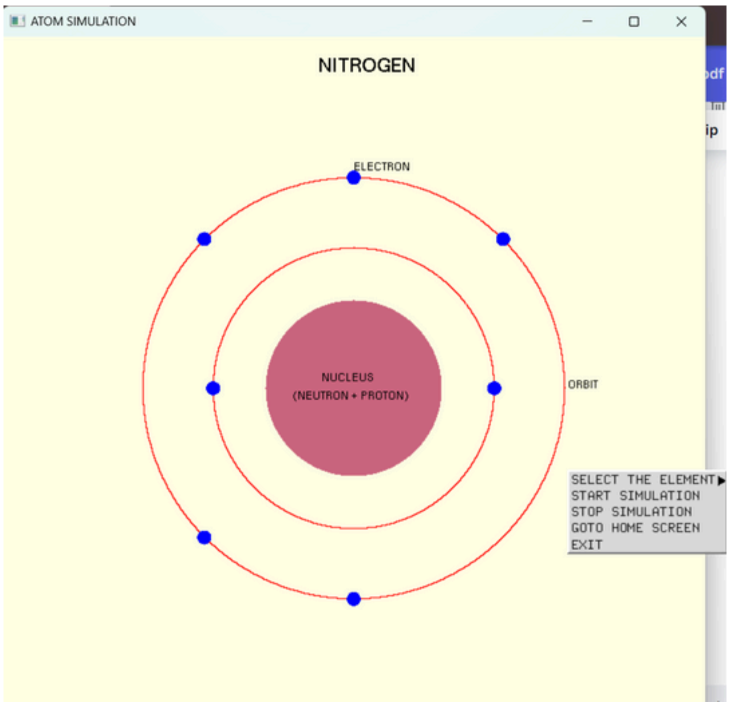
GUIDED BY:
MRS. CHETNA NEMADE

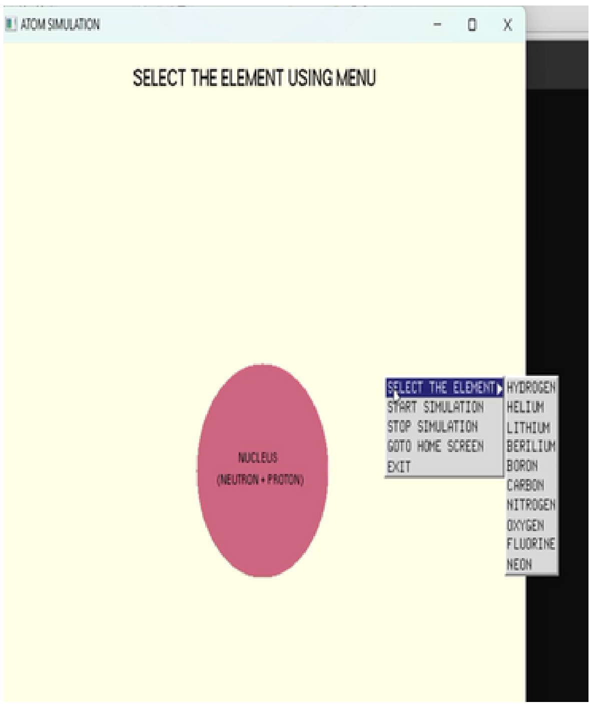
Press enter to Continue

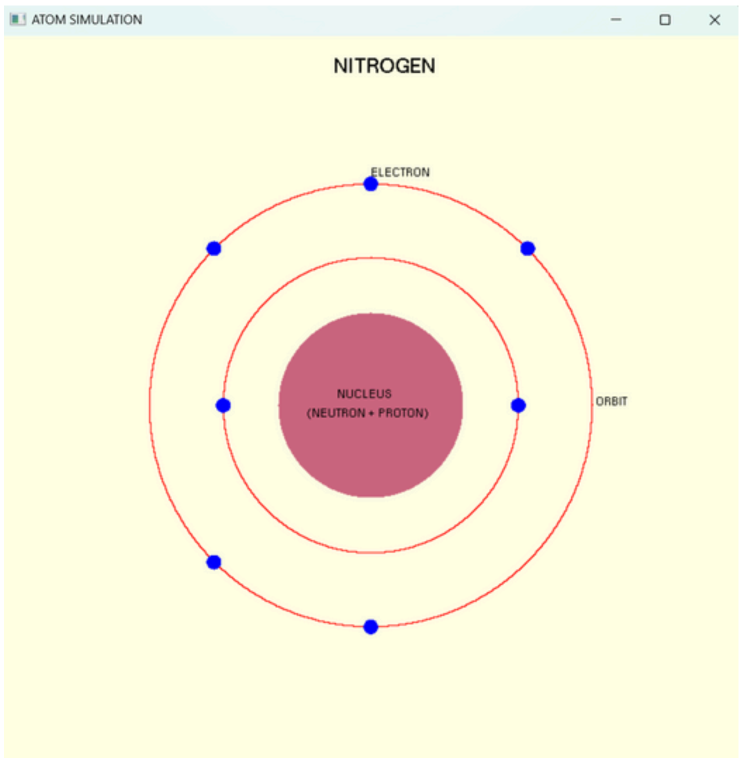


ATOM SIMULATION

SELECT THE ELEMENT USING MENU


ATOM SIMULATION
NITROGEN
ELECTRON
NUCLEUS
(NEUTRON + PROTON)
ORBIT
SELECT THE ELEMENT
START SIMULATION
STOP SIMULATION
GOTO HOME SCREEN
EXIT


ATOM SIMULATION
SELECT THE ELEMENT USING MENU
NUCLEUS
(NEUTRON + PROTON)
SELECT THE ELEMENT
START SIMULATION
STOP SIMULATION
GOTO HOME SCREEN
EXIT
HYDROGEN
HELIUM
LITHIUM
BERILIUM
BORON
CARBON
NITROGEN
OXYGEN
FLUORINE
NEON


ATOM SIMULATION
NITROGEN
ELECTRON
NUCLEUS
(NEUTRON + PROTON)
ORBIT