Projeto Final {

[Modelo de átomo de oxigênio]

< Efrain Marcelo >
< Fernando Lucas >
< Teophilo Vitor >

}

Sumário {

01 Ideia

< Motivação >

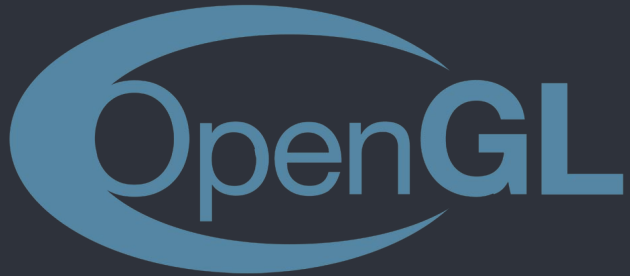
02 Desenvolvimento

< Código em OpenGL >

03 Resultado

< Apresentação do modelo final >

}

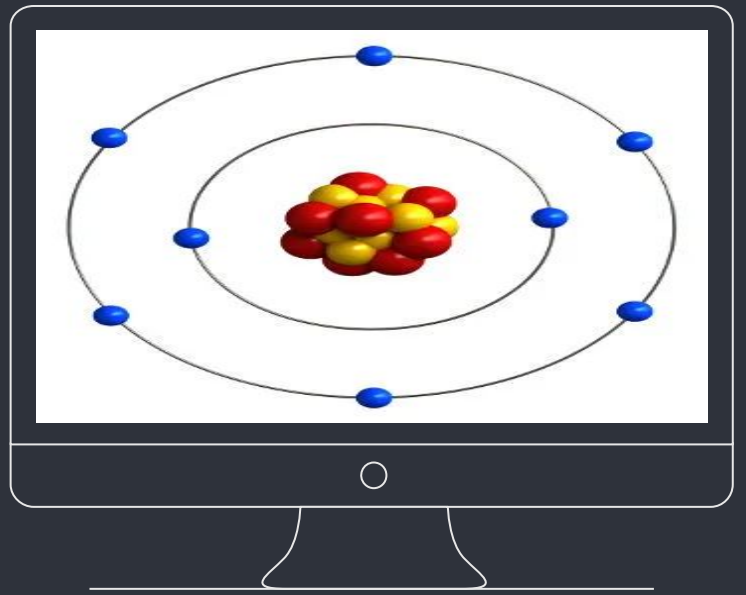


01 {

[Ideia]

< átomo >
< oxigênio >
< prótons >
< nêutrons >
< elétrons >

}

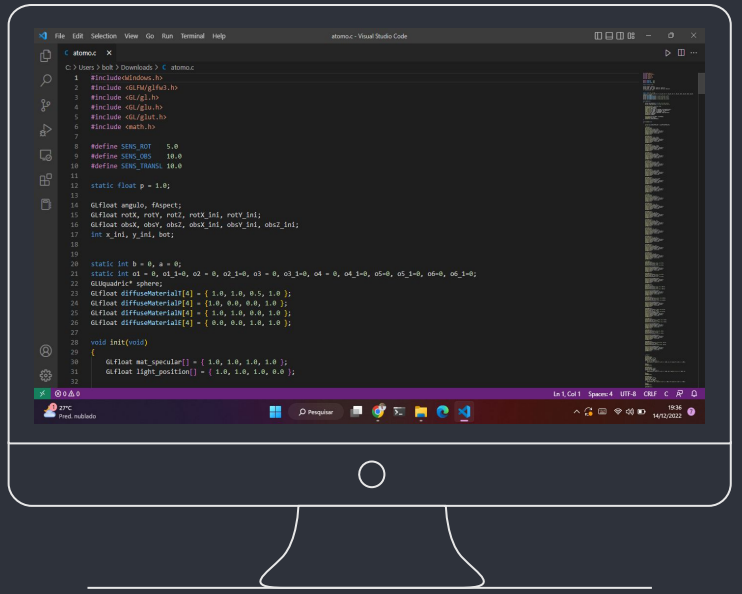


02 {

[Desenvolvimento]

- < openGL >
- < esferas >
- < cores >
- < câmera >
- < rotação >
- < translação >

}



DCA

parâmetros {

```
#define SENS_ROT    5.0
#define SENS_OBS    10.0
#define SENS_TRANSL 10.0

static float p = 1.0;

GLfloat angulo, fAspect;
GLfloat rotX, rotY, rotZ, rotX_ini, rotY_ini;
GLfloat obsX, obsY, obsZ, obsX_ini, obsY_ini, obsZ_ini;
int x_ini, y_ini, bot;

static int b = 0, a = 0;
static int o1 = 0, o1_1=0, o2 = 0, o2_1=0, o3 = 0, o3_1=0, o4 = 0, o4_1=0, o5=0, o5_1=0, o6=0, o6_1=0;
GLUquadric* sphere;
GLfloat diffuseMaterialP[4] = {1.0, 0.0, 0.0, 1.0 };
GLfloat diffuseMaterialN[4] = { 1.0, 1.0, 0.0, 1.0 };
GLfloat diffuseMaterialE[4] = { 0.0, 0.0, 1.0, 1.0 };
```

}

```
1
2   init{
3
4
5
6
7
8
9
10
11
12
13  }
14
```

```
void init(void)
{
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_DEPTH_TEST);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterialP);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 25.0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);

    glColorMaterial(GL_FRONT, GL_DIFFUSE);
    glShadeModel(GL_FLAT);
    glEnable(GL_COLOR_MATERIAL);
}
```

protóns {

```
}
```

```
//Prótons
glPushMatrix();
glTranslatef(0, 2.5, 2);
glColor4fv(diffuseMaterialP);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();

glPushMatrix();
glTranslatef(0, -2.5, 2);
glColor4fv(diffuseMaterialP);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();
```

nêutrons {

```
//Neutrons
glPushMatrix();
glTranslatef(0, 5.5, -2.0);
glColor4fv(diffuseMaterialN);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();

glPushMatrix();
glTranslatef(0, -5.5, -2.0);
glColor4fv(diffuseMaterialN);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();
```

}

```
elétrons{
}
```

```
//Elétrons
glPushMatrix();
glRotatef((GLfloat)o1, 0.0, 1.0, 0.0);
glTranslatef(15, 0.0, 0.0);
glRotatef((GLfloat)o2, 0.0, 1.0, 0.0);

glColor4fv(diffuseMaterialE);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();

glPushMatrix();
glRotatef((GLfloat)o1, 0.0, 1.0, 0.0);
glTranslatef(-15, 0.0, 0.0);
glRotatef((GLfloat)o2, 0.0, 1.0, 0.0);

glColor4fv(diffuseMaterialE);
gluQuadricDrawStyle(sphere, GLU_FILL);
gluQuadricNormals(sphere, GLU_FLAT);
gluSphere(sphere, 2, 50, 50);
glPopMatrix();
```

órbitas

```
//Orbitas
int i;
glPushMatrix();
glTranslated(0, 0, 0);
glRotatef(20, 0, 1, 0);
glBegin(GL_LINE_LOOP);
for (i = 0; i < 100; i++) {
    glVertex3f(15 * cos(2.0 * 3.14 * i / 100), 0, 15 * sin(2.0 * 3.14 * i / 100));
}
glEnd();
glPopMatrix();

glPushMatrix();
glTranslated(0, 0, 0);
glBegin(GL_LINE_LOOP);
for (i = 0; i < 100; i++) {
    glVertex3f(30 * sin(2.0 * 3.14 * i / 100), 30 * cos(2.0 * 3.14 * i / 100), 0);
}
glEnd();
glPopMatrix();
```

reshape{

```
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(90.0, (GLfloat)w / (GLfloat)h, 1.0, 200.0); //angulo
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 10.0, 20.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // visao da camera
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);      //melhora a qualidade dos gráficos
}
```

}

observador{

```
}
```

```
void PosicionaObservador(void)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    // Posiciona e orienta o observador
    glTranslatef(-obsX * 10, -obsY * 10, -obsZ * 10);
    glRotatef(rotX, 1, 0, 0);
    glRotatef(rotY, 0, 1, 0);
    glRotatef(rotZ, 0, 0, 1);
}
```

mouse{

```
}
```

```
void GerenciaMouse(int button, int state, int x, int y)
{
    if (state == GLUT_DOWN)
    {
        // Salva os parâmetros atuais
        x_ini = x;
        y_ini = y;
        obsX_ini = obsX;
        obsY_ini = obsY;
        obsZ_ini = obsZ;
        rotX_ini = rotX;
        rotY_ini = rotY;
        bot = button;
    }
    else bot = -1;
}
```

mouse{

```
}
```

```
void GerenciaMovim(int x, int y)
{
    // Botão esquerdo do mouse
    if (bot == GLUT_LEFT_BUTTON)
    {
        // Calcula diferenças
        int deltax = x_ini - x;
        int deltay = y_ini - y;
        // E modifica ângulos
        rotY = rotY_ini - deltax / SENS_ROT;
        rotX = rotX_ini - deltay / SENS_ROT;
    }
    // Botão direito do mouse
    else if (bot == GLUT_RIGHT_BUTTON)
    {
        int deltax = x_ini - x;
        int deltay = y_ini - y;
        // Calcula diferença
        int deltaz = deltax - deltay;
        // E modifica distância do observador
        obsZ = obsZ_ini + deltaz / SENS_OBS;
    }
```

```
    // Botão do meio
    else if (bot == GLUT_MIDDLE_BUTTON)
    {
        // Calcula diferenças
        int deltax = x_ini - x;
        int deltay = y_ini - y;
        // E modifica posições
        obsX = obsX_ini + deltax / SENS_TRANSL;
        obsY = obsY_ini - deltay / SENS_TRANSL;
    }
    PosicionaObservador();
    glutPostRedisplay();
}
```

keyboard{

```
}
```

```
void keyboard(unsigned char key, int x, int y)
{
    switch (key) {
        case 'a':
            o1 = (o1 + 12) % 360;
            o2 = (o2 + 12) % 360;
            o1_1 = (o1_1 - 12) % 360;
            o2_1 = (o2_1 - 12) % 360;
            o3 = (o3 - 30) % 360;
            o4 = (o4 - 30) % 360;
            o3_1 = (o3_1 + 30) % 360;
            o4_1 = (o4_1 + 30) % 360;
            o5 = (o5 - 15) % 360;
            o6 = (o6 - 15) % 360;
            o5_1 = (o5_1 + 15) % 360;
            o6_1 = (o6_1 + 15) % 360;
            glutPostRedisplay();
            break;
        default:
            break;
    }
}
```

```
main{
}
```

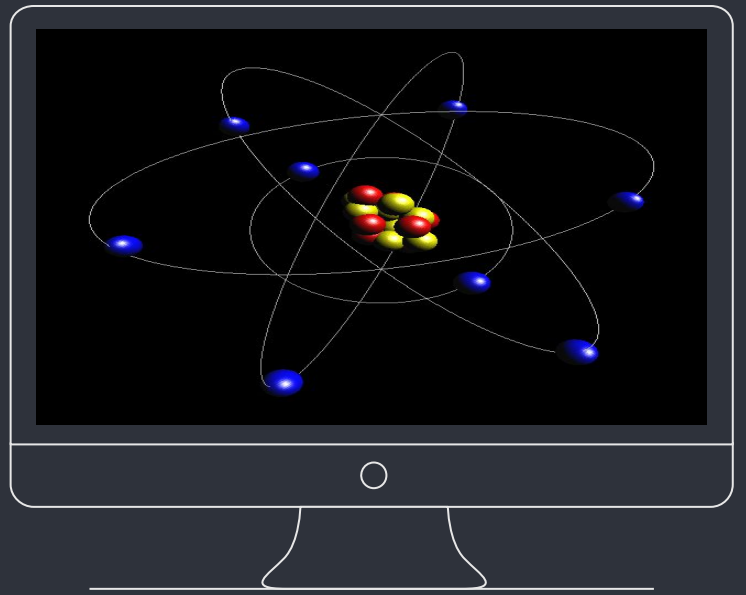
```
int main(int argc, char** argv)
{

    glEnable(GL_DEPTH_TEST);
    sphere = gluNewQuadric();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1920, 1080);
    glutInitWindowPosition(100, 100);
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutMouseFunc(GerenciaMouse);
    glutMotionFunc(GerenciaMovim);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

03 {

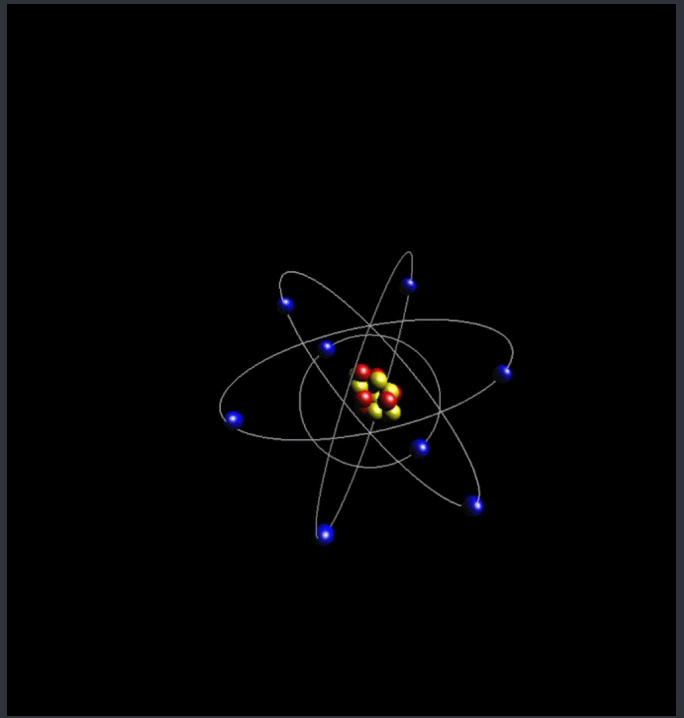
[Resultado]

< átomo >
< oxigênio >
< eletrosfera >
< núcleo >
< modelo >

}



Modelo final {



}

```
FIM {
Obrigado;

}
```