Universidade Federal de Lavras
Departamento de Ciência da Computação

escavadeira.cpp v2.0

Projeto Prático da disciplina GCC124 - Computação Gráfica

Integrantes:

Mateus Carvalho Gonçalves - 201810245
Sérgio Henrique Menta Garcia -201810013

Turma 10A

Introdução

- escavadeira.cpp: projeto de uma escavadeira 3D interativa e animada.
- Desenvolvido em C++ utilizando as APIs OpenGL, GLU e GLUT.

Objetivos

Aplicar os conteúdos apresentados durante a disciplina GCC124 - Computação Gráfica, com foco nas seguintes exigências:

1. Desenvolvimento de ambiente 3D em OpenGL;
2. Ambiente colorido - cores diferentes para objetos distintos;
3. Interação com teclado e mouse;
4. Operações de transformações geométricas;
5. Iluminação - permitindo ligar/desligar luz;
6. Animação de pelo menos 1 objeto;
7. Textura em pelo menos 1 objeto.

Para atingir os objetivos, uma série de requisitos para o projeto foram definidos.

Requisitos

Os requisitos definidos pelo grupo resultaram nas animações desenvolvidas, combinando as transformações geométricas com o mapeamento do teclado e mouse, e eles são:

COMANDO	DESCRIÇÃO
‘Q’ / ‘q’	Gira toda parte superior da escavadeira, nos dois sentidos, por meio da rotação da ‘cintura’
‘W’ / ‘w’	Movimenta, nos dois sentidos, o ‘braço’ da escavadeira por meio da rotação do ‘ombro’
‘E’ / ‘e’	Movimento, nos dois sentidos, o ‘antebraço’ da escavadeira por meio da rotação do ‘cotovelo’
‘R’ / ‘r’	Movimenta, nos dois sentidos, a pá da escavadeira por meio da rotação do ponto que liga a pá ao antebraço

Requisitos

COMANDO	DESCRIÇÃO
‘A’ / ‘a’	Gira as 4 rodas simultaneamente, nos dois sentidos de rotação
‘+’ / ‘-’	Zoom in e zoom out
<space> / <backspace>	Movimenta a escavadeira para frente e para trás, e gira as 4 rodas simultaneamente, nos dois sentidos de rotação
‘↑’ / ‘↓’	Rotaciona o plano em torno do eixo x, nos dois sentidos
‘←’ / ‘→’	Rotaciona o plano em torno do eixo y, nos dois sentidos
<mouse left button>	‘Liga’ / ’Desliga’ luz e troca cor do farol

Principais referências

Documentação do OpenGL Khronos. Disponível em https://www.khronos.org/opengl/

OpenGL. Disponível em https://www.opengl.org/

PUCRS. Introdução à OpenGL. Disponível em https://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html

Sergio Silva. OpenGL C++ - Gráficos 3D. Disponível em https://www.youtube.com/playlist?list=PLVRDPs83ZhmcXYuktF3r2hfyoabg_EVPO

Codificação

```
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGBA);
    glutInitWindowSize (700, 500);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("Escavadeira v2.0");
    init();
    glutReshapeFunc(reshape);
    glutDisplayFunc(renderScene);
    glutMouseFunc(mouseEvents);
    glutKeyboardFunc(keyboardEvents);
    glutSpecialFunc(specialKeysEvents);
    glutMainLoop();

    return 0;
}
```

Interação e animação

```
// definicoes ascii
#define BACKSPACE 8
#define ESC       27
#define SPACE     32
#define PLUS      43
#define MINUS     45

// variaveis de manipulacao de acoes
GLfloat zoom = 0, run = 0.0;
static int xAngle = 0, yAngle = 0;
static int wheels = 0, waist = 0, shoulder = 0, elbow = 0, shovel = 25;
static bool farol = false;
```

Interação e animação

```
        case 'w':
            if(shoulder <= 75)
                shoulder += 1;
            break;
        case 'W':
            if(shoulder >= 0)
                shoulder -= 1;
            break;
        case 'e':
            if(elbow <= 0)
                elbow += 1;
            break;
        case 'E':
            if(elbow >= -75)
                elbow -= 1;
            break;
        case 'r':
            if(shovel <= 55)
                shovel += 1;
            break;
        case 'R':
            if(shovel >= -5)
                shovel -= 1;
            break;
```

Interação e animação

```
// definicao dos eventos de teclado (teclas especiais)
void specialKeysEvents(int key, int x, int y) {
    switch(key) {
        case GLUT_KEY_DOWN:
            if(xAngle > -10) {
                xAngle = xAngle - 5;
            }
            break;
        case GLUT_KEY_UP:
            if(xAngle < 90) {
                xAngle = xAngle + 5;
            }
            break;
        case GLUT_KEY_RIGHT:
            yAngle = (yAngle - 5) % 360;
            break;
        case GLUT_KEY_LEFT:
            yAngle = (yAngle + 5) % 360;
            break;
        default:
            break;
    }

    glutPostRedisplay();    // marca matriz como necessario de redesenhar
}
```

escavadeira.cpp v2.0

Interação e animação

```
// definicao dos eventos de mouse
void mouseEvents(int button, int state, int x, int y) {
    if (button == GLUT_LEFT_BUTTON){
        if(state == GLUT_UP) {
            if(!farol)
                farol = true;
            else
                farol = false;
        }
    }

    glutPostRedisplay();    // marca matriz como necessario de redesenhar
}
```

Iluminação

```
// variaveis de iluminação
GLfloat darkMode[] = { 0.1f, 0.1f, 0.1f, 1.0f };
GLfloat ligthAmbient[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat lightDiffuse[] = { 0.8f, 0.8f, 0.8f, 1.0f };
GLfloat lightSpecular[] = { 1.0f, 1.0f, 1.0f, 1.0f };
GLfloat lightPosition[] = { 2.0f, 5.0f, 5.0f, 0.0f };
```

Iluminação

```
void light() {
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_LIGHTING);

    if(farol == true) {
        glLightfv(GL_LIGHT0, GL_AMBIENT, ligthAmbient);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
        glLightfv(GL_LIGHT0, GL_SPECULAR, lightDiffuse);
        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
    } else {
        glLightfv(GL_LIGHT0, GL_AMBIENT, darkMode);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, darkMode);
        glLightfv(GL_LIGHT0, GL_SPECULAR, darkMode);
        glLightfv(GL_LIGHT0, GL_POSITION, lightPosition);
    }
}
```

Texturização

```
/* Cria textura de tabuleiro de xadrez */
#define checkeredBorderWidth 16
#define checkeredBorderHeight 16
static GLubyte checkeredBorder[checkeredBorderHeight][checkeredBorderWidth][4];

#ifdef GL_VERSION_1_1
    static GLuint texName;
#endif

void makecheckeredBorder(void) {
   int i, j, c;

   for(i = 0; i < checkeredBorderHeight; i++) {
        for (j = 0; j < checkeredBorderWidth; j++) {
            c = ((((i&0x8)==0)^((j&0x8))==0))*255;
            checkeredBorder[i][j][0] = (GLubyte) c;
            checkeredBorder[i][j][1] = (GLubyte) c;
            checkeredBorder[i][j][2] = (GLubyte) c;
            checkeredBorder[i][j][3] = (GLubyte) 255;
        }
   }
}
/* ---------------------------------- */
```

Texturização

```
makecheckeredBorder();
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

#ifdef GL_VERSION_1_1
    glGenTextures(1, &texName);
    glBindTexture(GL_TEXTURE_2D, texName);
#endif

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

#ifdef GL_VERSION_1_1
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkeredBorderWidth, checkeredBorderHeight,
                 0, GL_RGBA, GL_UNSIGNED_BYTE, checkeredBorder);
#else
    glTexImage2D(GL_TEXTURE_2D, 0, 4, checkeredBorderWidth, checkeredBorderHeight,
                 0, GL_RGBA, GL_UNSIGNED_BYTE, checkeredBorder);
#endif
```

Texturização

```
void drawAsphaltBorder(float halfWidth, float halfLength, float asphaltHalfLength) {
    glEnable(GL_TEXTURE_2D);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
    #ifdef GL_VERSION_1_1
        glBindTexture(GL_TEXTURE_2D, texName);
    #endif

    glTranslatef(-1000, 0.0, 0.0);
    glPushMatrix();

        for(int i = 0; i < (2 * asphaltHalfLength); i += (2 * halfLength)) {
            glBegin(GL_QUADS);
                glTexCoord2f(0.0, 0.0); glVertex3f(-halfLength, 0.0, -halfWidth);
                glTexCoord2f(1.0, 0.0); glVertex3f(halfLength, 0.0, -halfWidth);
                glTexCoord2f(1.0, 1.0); glVertex3f(halfLength, 0.0, halfWidth);
                glTexCoord2f(0.0, 1.0); glVertex3f(-halfLength, 0.0, halfWidth);
            glEnd();
            glTranslatef((2 * halfLength), 0.0, 0.0);
        }
    glPopMatrix();

    glDisable(GL_TEXTURE_2D);
}
```

OBRIGADO!