Projeto 1: Cena renderizada com pipeline convencional

Lucas Simões de Almeida – 1712101

Neste projeto, tivemos que renderizar uma cena utilizando um pipeline convencional do openGL, esta cena deve ter obrigatoriamente: uma mesa, instâncias de cubos e esferas, uma folha quadricular sobre a mesa, e duas fontes de luz, uma do ponto de visualização e outra da luminária.

Esta tarefa foi especialmente desafiadora, pois precisaríamos criar diversos objetos geométricos, e praticamente cada um deles precisaria de funções especializadas para sua criação, mas por exemplo a função do cubo pode ser utilizada através de transformações para criar outros tipos de quadrilátero, como a mesa, os pés da mesa e até a folha de papel.

```
//Desenha o cubo
glPushMatrix();
glScalef(0.3f, 0.3f, 0.3f);
glTranslatef(0.0f, 1.0f, 0.f);
glRotatef(theta2, 0.0f, 1.0f, 0.0f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, blue);
drawCube();
glPopMatrix();

//desenha a mesa
glPushMatrix();
glScalef(3.5f, 0.3f, 3.5f);
glTranslatef(0.0f, 0.0f, 0.0f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, white);
drawCube();
glPopMatrix();

//Desenha pe direito inferior da mesa
glPushMatrix();
glScalef(0.3f, 2.0f, 0.3f);
glTranslatef(3.5f, -0.5f, 3.8f);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
drawCube();
glPopMatrix();
```

A parte mais desafiadora com certeza foi o desenvolvimento do controle de câmera ArcBall, utilizando de dois vetores, correspondendo a posições do mouse, precisaríamos bolar uma lógica para rotacionar e transladar corretamente nosso desenho.

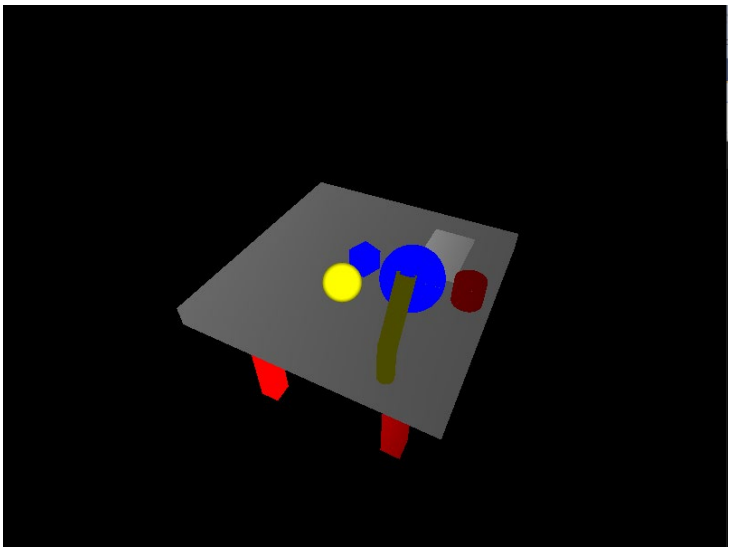
```
static void motionCB(int x, int y) {
    float a[3] = { 0.0f,0.0f,0.0f };
    float d = viewer_pos[2];
    float r = fmin(w_s, h_s) / 2;
    // x1, y1 --> X1, Y1, Z1
    X1 = (float(x_0) - (w_s / 2)) / r;
    Y1 = (float(y_0) - (h_s / 2)) / r;
    dist1 = sqrt((1-(X1*X1 + Y1*Y1)));
    // x, (h - y) --> X2, Y2, Z2
    y = h_s - y;
    X2 = ((float)x - (w_s / 2)) / r;
    Y2 = ((float)y - (h_s / 2)) / r;
    dist2 = sqrtf(X2 * X2 + Y2 * Y2);
    if (dist1 > 1)
    {
        X1 = X1 / dist1;
        Y1 = Y1 / dist1;
        Z1 = 0.0f;
    }
    else
    {
        dist1 = sqrt(X1 * X1 + Y1 * Y1);
        dist1 = (1 - (dist1 * dist1));
        Z1 = sqrt(dist1);
    }
```

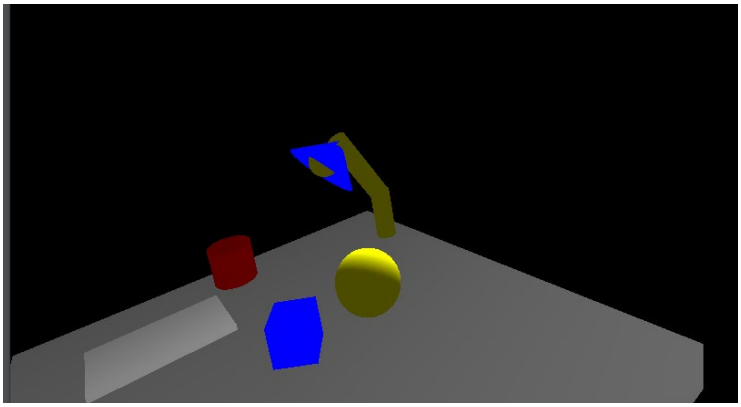
```
if (dist2 > 1)
{
    X2 = X2 / dist2;
    Y2 = Y2 / dist2;
    Z2 = 0.0f;
}
else
{

    dist2 = sqrt(X2 * X2 + Y2 * Y2);
    dist2 = (1 - (dist2 * dist2));
    Z2 = sqrtf(dist2);

}

// u, v --> a, theta
u[0] = X1;
u[1] = Y1;
u[2] = Z1;

v[0] = X2;
v[1] = Y2;
v[2] = Z2;
vet_product(u, v, a);
float theta = acos(sqrtf(powf(a[0], 2) + powf(a[1], 2) + powf(a[2], 2)));

glPushMatrix();
glLoadIdentity();
glTranslatef(0.0f, 0.0f, -d);
glRotatef(theta, a[0], a[1], a[2]);
glTranslatef(0.0f, 0.0f, d);
glMultMatrixf(M);
glGetFloatv(GL_MODELVIEW_MATRIX, M);
glPopMatrix();

//float dist=sqrtf(dist1 * dist1 + dist2 * dist2);
x_0 = (float)x;
y_0 = (float)y;

glutPostRedisplay();
```

Seguindo os slides de aula, tive uma boa ideia de como começar, porém diversos problemas surgiam durante seu desenvolvimento, ele na maioria das vezes com um clique do mouse rotacionava ou transladava tão rápido e com tamanha intensidade que o desenho só desaparecia da tela.

Eu acabei optando por usar o arc cosseno no calculo de theta, pois fazia mais sentido com a logica apresentada, o que me poupou um tremendo esforço.

Após a definição do controle de câmera, continuei a criação do desenho, o que eventualmente resultou na imagem abaixo:





Deste ângulo, podemos observar a lâmpada da luminária.

E por último, criar as nossas fontes de iluminação, esse passo eu sinceramente não sei dizer se fiz corretamente, as fontes de luz foram criadas, porém a fonte da luz da iluminaria parece muito fraca, eu tentei diversas funções para intensificar sua luz mas em vão.

```
GLfloat spot_direction[] = { -1.00f, -1.00f, 0.00f  };
//Definindo fontes de luz
float amb[] = { 0.1f , 0.1f , 0.1f , 1.0f };
float dif[] = { 0.6f , 0.6f ,0.6f , 1.0f };
float amb2[] = { 0.2f , 0.2f , 0.2f , 1.0f };
float dif2[] = { 0.8f , 0.8f ,0.8f , 1.0f };
glLightfv(GL_LIGHT0, GL_POSITION, viewer_pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif);

glLightfv(GL_LIGHT1, GL_POSITION, luzLampada);
glLightfv(GL_LIGHT1, GL_AMBIENT, amb2);
glLightfv(GL_LIGHT1, GL_DIFFUSE, dif2);
glLightfv(GL_LIGHT1, GL_SPECULAR, dif2);
glLightfv(GL_LIGHT1, GL_SPOT_DIRECTION, spot_direction);
glLightf(GL_LIGHT1, GL_SPOT_CUTOFF, 60.0f);
//glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 100.0f);
```

Como podemos ver na imagem acima, a fonte de luz da luminária possui um posicionamento específico que coincide com o desenho da lâmpada, também possui um direcionamento e um limite do seu ângulo de iluminação.

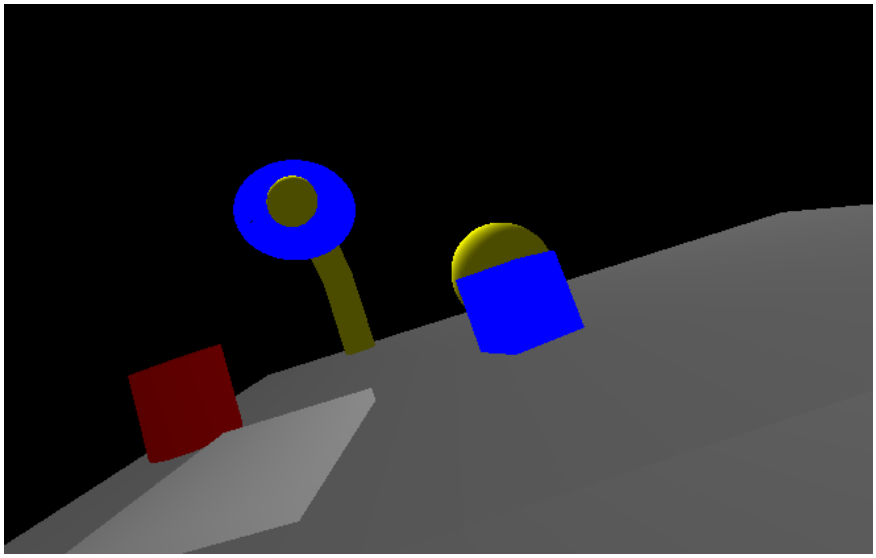


Imagem capturada para demonstrar a flexibilidade do controle de câmera ArcBall