Computação Gráfica — Trabalho 3

Professor: Waldemar Celes
Aluno: Antenor Barros Leal

01 de dezembro de 2024

1 Resumo

Este trabalho tem como objetivo fazer a implementação e teste de técnicas de renderização em uma cena 3D.

2 Cena Base

Para a cena utilizou-se uma versão com leves modificações a partir da tarefa 2.1.

Para este trabalho o cubo cinza foi retirado para ser substituído por um quad que fará papel de uma superfície plana para o refletor e como anteparo para a projeção das sombras.

O seguinte grafo de cena foi usado:

```
Node::Make(shader,
    {
        Node::Make(trCube2,{yellow},{cube}),
        Node::Make(trSphere1,{green},{sphere}),
        Node::Make(trSphere2,{red},{sphere}),
        Node::Make(trCube3,{red},{cube}),
    }
);
```

3 Técnicas de Renderização

Entre as opções, foi escolhida a técnica de reflexão planar e de sombra planar.

3.1 Técnica: reflexão planar

Como dito na seção anterior foi usado um quad para "receber" a reflexão que é simplesmente a repetição da cena de $y > 0$ em $y < 0$ com a componente y com sinal trocado. Usando o transformador de escala, conseguimos fazer esta inversão vertical:

```
trf->Scale(1.0f,-1.0f,1.0f);
```

Todavia, se apenas isto for feito, a reflexão irá "vazar" para fora da superfície reflexiva.

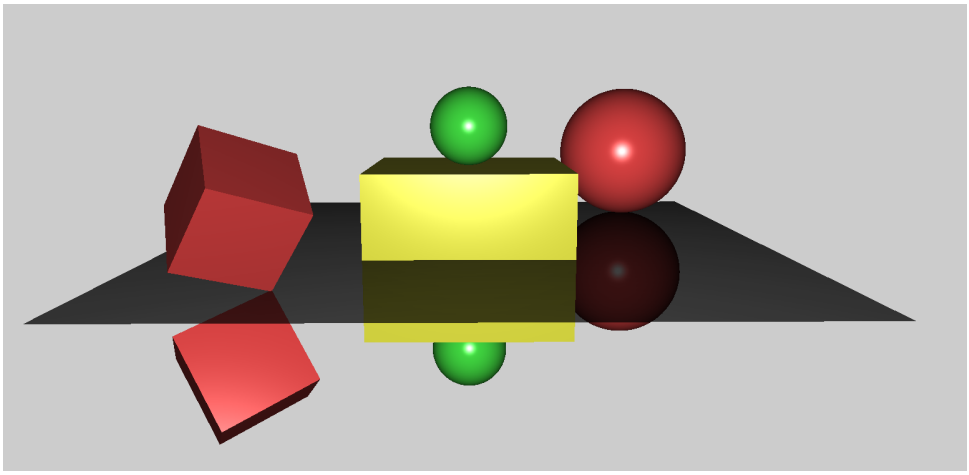

Figure 1: Antes do stencil

Isto é resolvido aplicando um stencil na superficie reflexiva e informando ao OpenGL não renderizar a cena que esteja fora desta máscara.

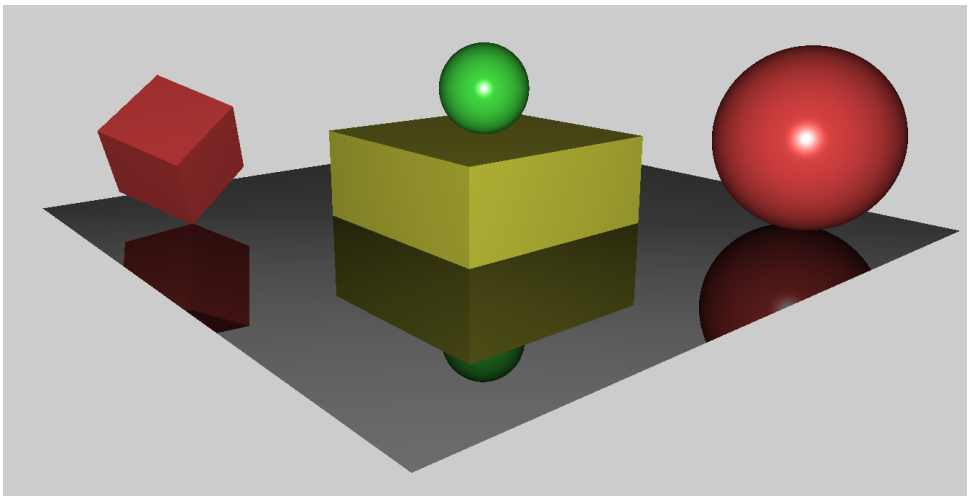

Figure 2: Depois do stencil

Porém se mudarmos o arcball para visualizar a parte de baixo da superficie reflexiva, vemos parte do reflexo cortado pelo stencil. O que se é desejado, obviamente, é a ausência de qualquer objeto.

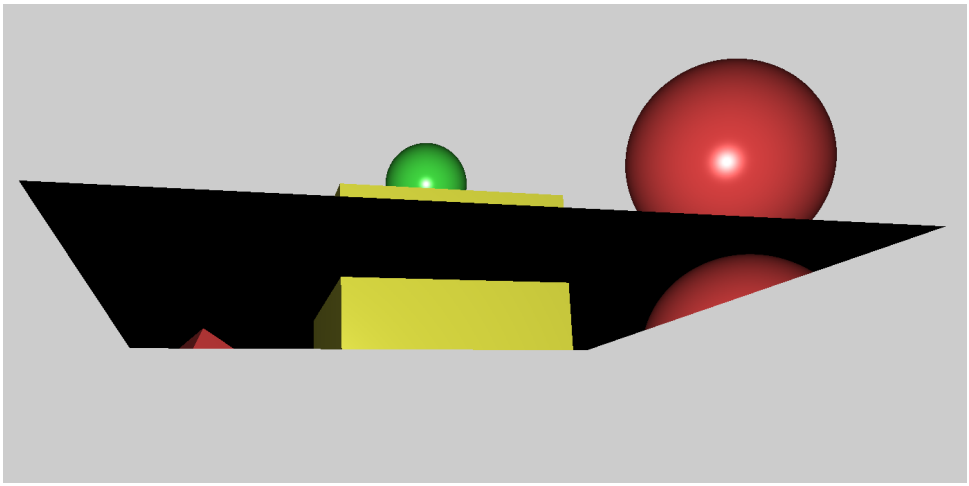

Figure 3: Cena refletida aparecendo do outro lado

Para resolver isso, normalmente é usado um plano de corte para mostrar a cena refletida apenas acima deste plano. Porém não foi conseguido realizar isto. Para contornar, na função de display é testado se a câmera está no lado de cima do refletor. Apenas se este for o caso, a cena refletida aparece.

```
if (camera->GetViewMatrix()[1][2] > 0) {
  ...
```

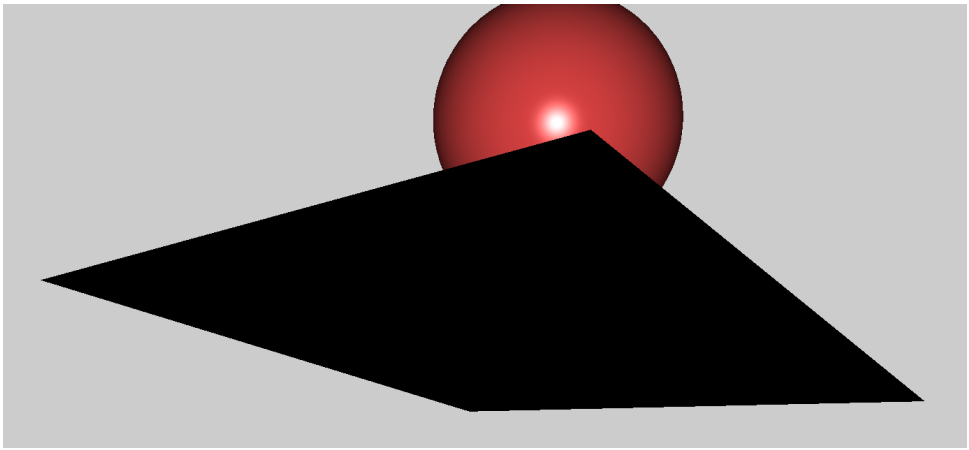

Figure 4: Cena corrigida

3.2 Técnica: sombra planar

Para a sombra planar, aplicamos a geometria em uma matriz de projeção que "achata" os objetos de cena em uma superfície 2D. Esta matriz de projeção, por considerar a posição da luz, consegue simular corretamente como é a sombra projetada por esta luz.

3.2.1 Matriz de projeção

- Seja n o vetor $[n_x n_y n_z]$, representando a normal ao plano onde a sombra deve ser exibida com os índices iguais às componentes x, y e z.
- Seja, l o vetor $[l_x l_y l_z]$ com a posição da fonte de luz.
- Seja p o vetor $[p_x p_y p_z 1]$ com a posição de um ponto na geometria e o último índice sempre igual a um.

Cada ponto da geometria representado pelo vetor quando for multiplicado pela matriz M.

$$M = \begin{bmatrix} nl + n_w - l_x n_x & -l_x n_y & -l_x n_z & -l_x n_w \\ -l_y n_x & nl + n_w - l_y n_y & -l_y n_z & -l_y n_w \\ -l_z n_x & -l_z n_y & nl + n_w - l_z n_z & -l_z n_w \\ -n_x & -n_y & -n_z & nl \end{bmatrix}$$

Será transformado em p', um vetor que representa um ponto no plano da sombra.

$$p' = Mp$$

Fazendo esta operação com todos os pontos da geometria, teremos todos os pontos da sombra.

3.2.2 Algoritmo

Para renderizar a cena, faz-se quatro passadas. Inicialmente é limpado o stencil porque se não teríamos um efeito de "pintura", já que as marcações do stencil iriam se acumular de quadro a quadro.

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);
```

Foi modificado o shader para que fosse possível renderizar a cena apenas com a luz ambiente caso uma variável fosse verdadeira.

```
if (amb_only) {
  fcolor = ambient;
}
else {
  fcolor = ambient + diffuse + specular;
}
```

Na primeira passada, é renderizado o chão apenas com a luz ambiente. Isto será a cor da sombra.

```
amb_only->SetValue(true);
sceneGround->Render(camera);
amb_only->SetValue(false);
```

Na segunda passada, a geometria da cena é multiplicada pela matriz de sombra e o stencil é marcado com a resultante. Esta marcação é a sombra.

```
glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_NEVER , 1, 0xFFFF);
glStencilOp(GL_REPLACE , GL_REPLACE , GL_REPLACE);
glm::mat4 sm = shadowMatrix(glm::vec4(0.0f,0.0f,1.0f,1.0f),glm::vec4(2.0f, 2.0f,
10.0f, 1.0f));
TransformPtr tr = Transform::Make();
tr->MultMatrix(sm);
sceneRoot->GetRoot()->SetTransform(tr);
sceneRoot->Render(camera);
sceneRoot->GetRoot()->SetTransform(nullptr);
```

Na terceira passada, o chão é desenhado normalmente, exceto pelas áreas onde o stencil foi marcado. Nestas áreas, apenas a luz ambiente do passo anterior irá aparecer.

```
glStencilFunc(GL_EQUAL , 0, 0xFFFF);
glStencilOp(GL_KEEP , GL_KEEP , GL_KEEP);
glBlendFunc(GL_ONE ,GL_ONE);
glEnable(GL_BLEND);
glDepthFunc(GL_EQUAL);
sceneGround->Render(camera);
glDepthFunc(GL_LESS);
glDisable(GL_STENCIL_TEST);
glDisable(GL_BLEND);
```

Por fim, a cena principal é desenhada normalmente.

```
sceneRoot->Render(camera);
```

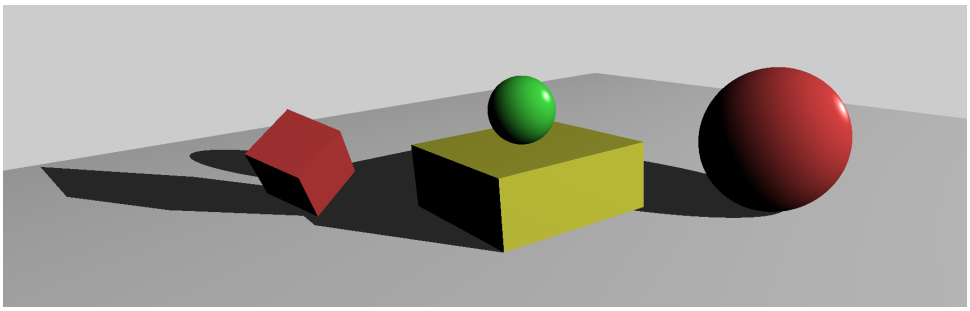

Figure 5: Cena com sombra

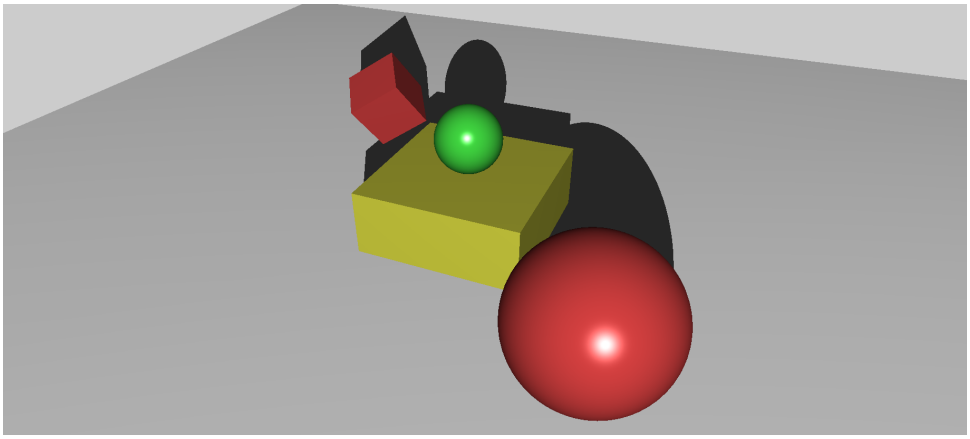

Figure 6: Cena com sombra 2

4 Resultados

Além das capturas de tela deste relatório, um vídeo de demostração do funcionamento foi incluído no arquivo "demo.mp4".