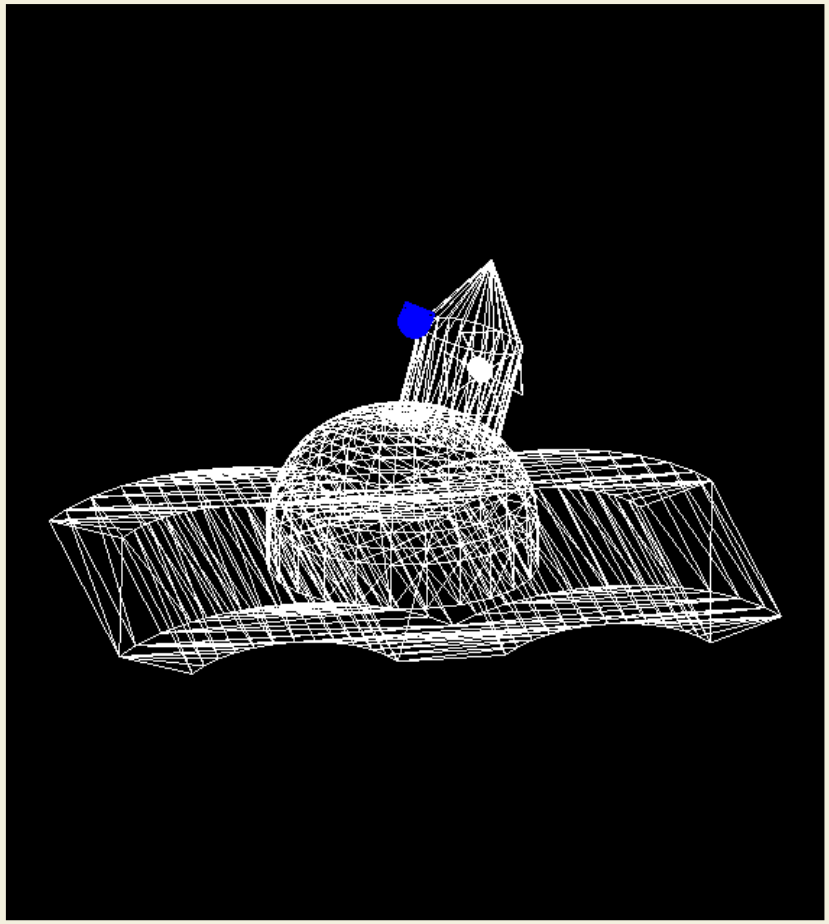
Trabalho Final de CG

Introdução - Objetivo

- Aplicar as noções de projeções e iluminação vistas em sala no modelo criado na primeira parte.

Relembrando o modelo

- Tema de fantasia, foi inspirado em livros de pop-up
- Feito no SketchUp Make
- Era assim...



Mudanças no Leitor de .obj

ColorMatrix

- Estrutura de dados que guarda uma matriz 3x3.
- Cada linha representa uma componente.
- É criado um objeto desse para cada material do objeto, cada face apenas referencia seu material.

ColorMatrix

- *float[3][3]*
- toColor()

Arquivo .mtl

```
# comentário
newmtl Moinho
Ka 1.000000 1.000000 1.000000
Kd 0.772549 0.741176 0.937255
Ks 0.330000 0.330000 0.330000
Ns 1
```

Leitura

- A leitura é bem parecida do modelo usando a mesma expressão regular, como mostrado no slide anterior.
- No fim temos um vetor associativo ligando cada *string* com seu material associado.
- Cada vez que o .obj requer um material, acessamos o vetor e guardamos a referencia do material na face.

Iluminação

Adaptando a renderização

- Para um modelo ser renderizado agora ele precisa saber quem são as luzes da cena.
- Como decidimos usar o *flat shading* temos que calcular o vetor normal e o ponto médio do triângulo.
- Já que cada face sabe seu material, decidir a cor desta face é só passar o ponto médio e a normal para o ColorMatrix.

```
void Model::desenhar(float ambiente[3],QList<Light> luzes);

float* ColorMatrix::toColor(Vértice point,Vetor normal,float
ambiente[3],QList<Light> luzes);
```

Por dentro do toColor()

- Implementamos a fórmula do somatório usual.

$$C := k_a I_a + \sum_{l \in Luzes} k_d l_{ka} (D_l \cdot N) + k_s I_{ks} (R_l \cdot V)$$

Projeções

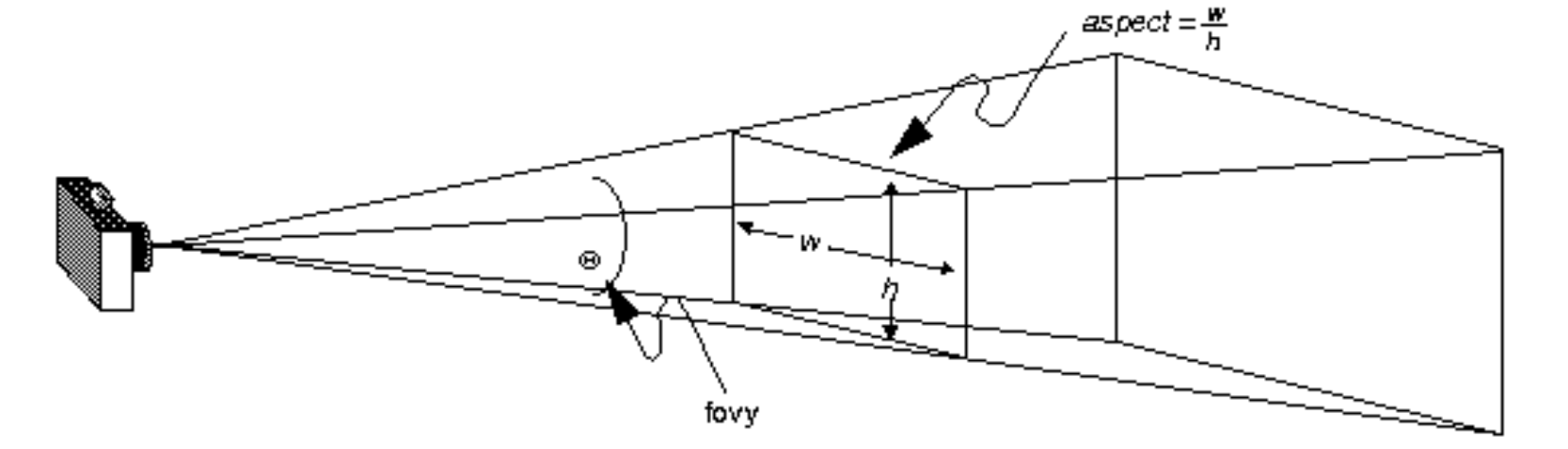
Matrizes de Projeção

- Funções que recebem os parâmetros desejados e retornam a matriz que transforma na projeção canônica
- O padrão do OpenGL usa a projeção ortográfica canônica

```
TransformMatrix ortho(left,right,bottom,up,near,far);
TransformMatrix frustum(left,right,bottom,up,near,far);
TransformMatrix perspective(fov,aspectRatio,near,far);
TransformMatrix isometric(scale,near,far,horizontal,vertical);
```

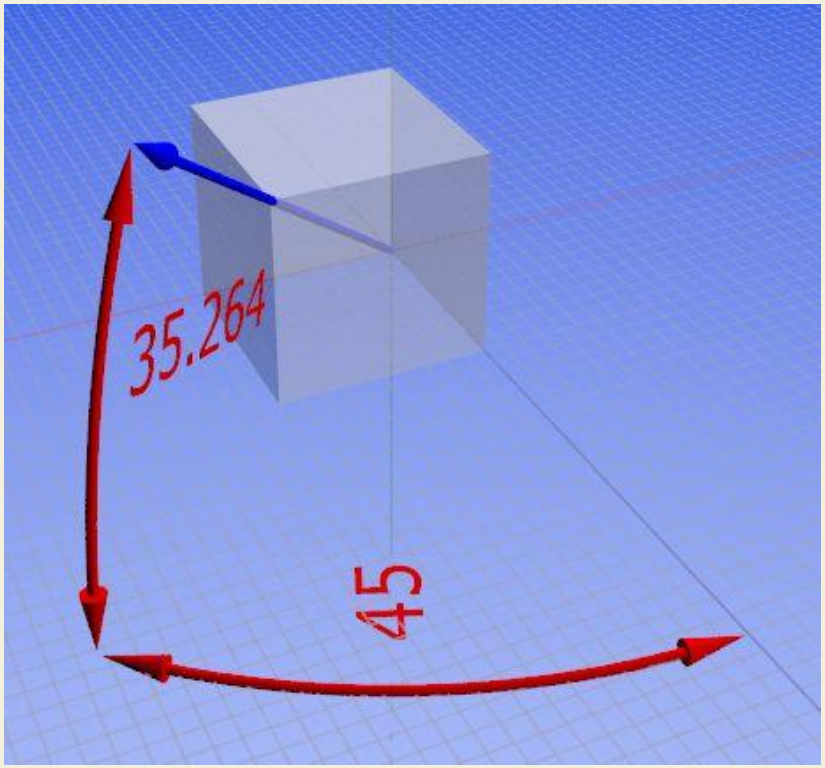
Perspectiva

- Implementa a função perspective do Glut.
- Recebe o ângulo visual da câmera, a proporção da imagem e a posição dos planos *near* e *far*.
- Equivalente ao frustrum.

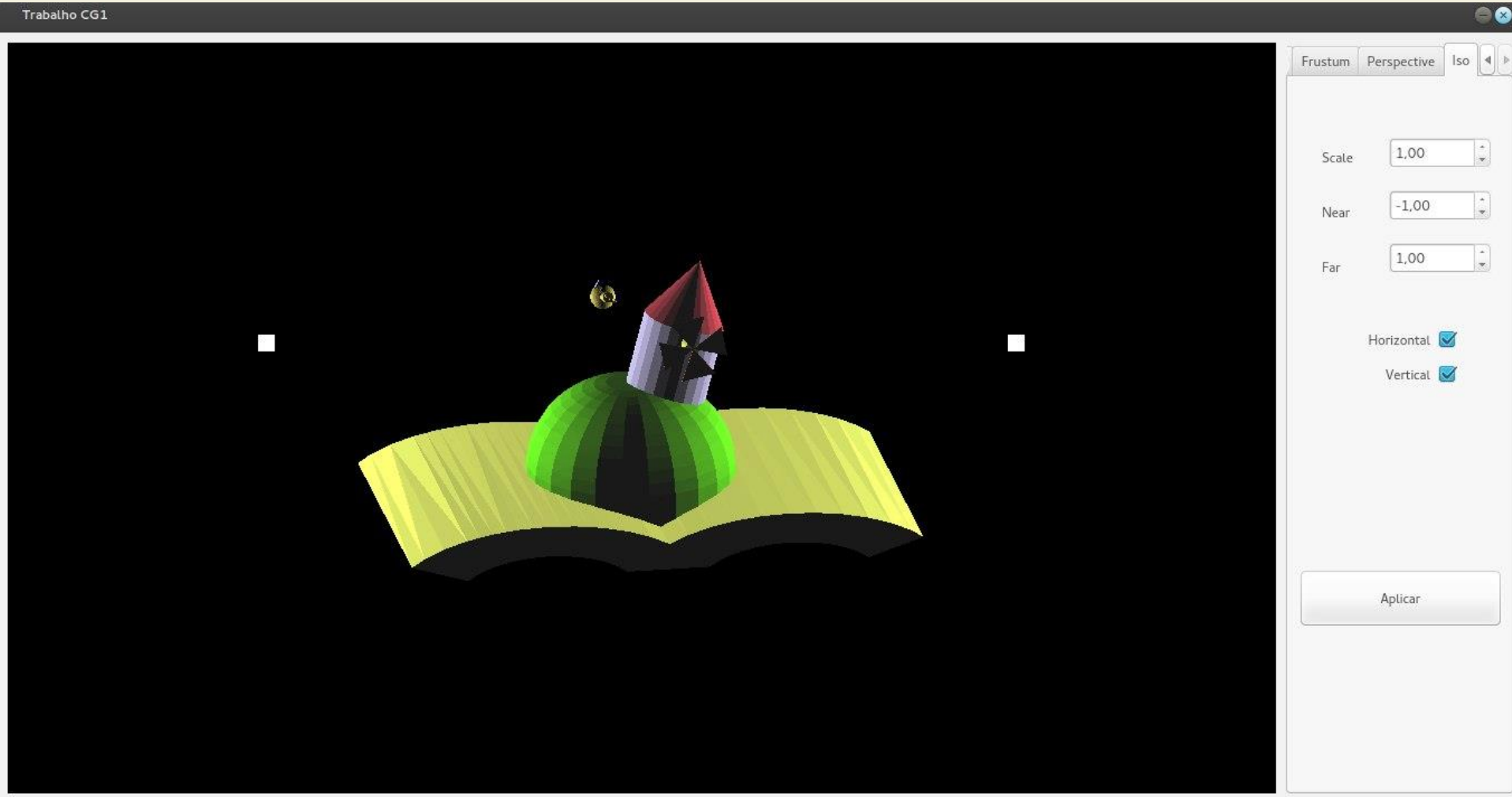


Isométrica

- Implementa a projeção isométrica.
- Apenas rotaciona a cena e aplica uma projeção ortográfica.
- Os dois booleanos indicam qual sentido será dada a rotação.



Interface para Projeções



Conclusão

- O trabalho foi uma experiência positiva para o aprendizado da matéria por utilizar a teoria dada em sala.
- O incentivo ao aprofundamento do uso de C++ foi intenso e gerou conhecimento prático útil para a vida profissional.