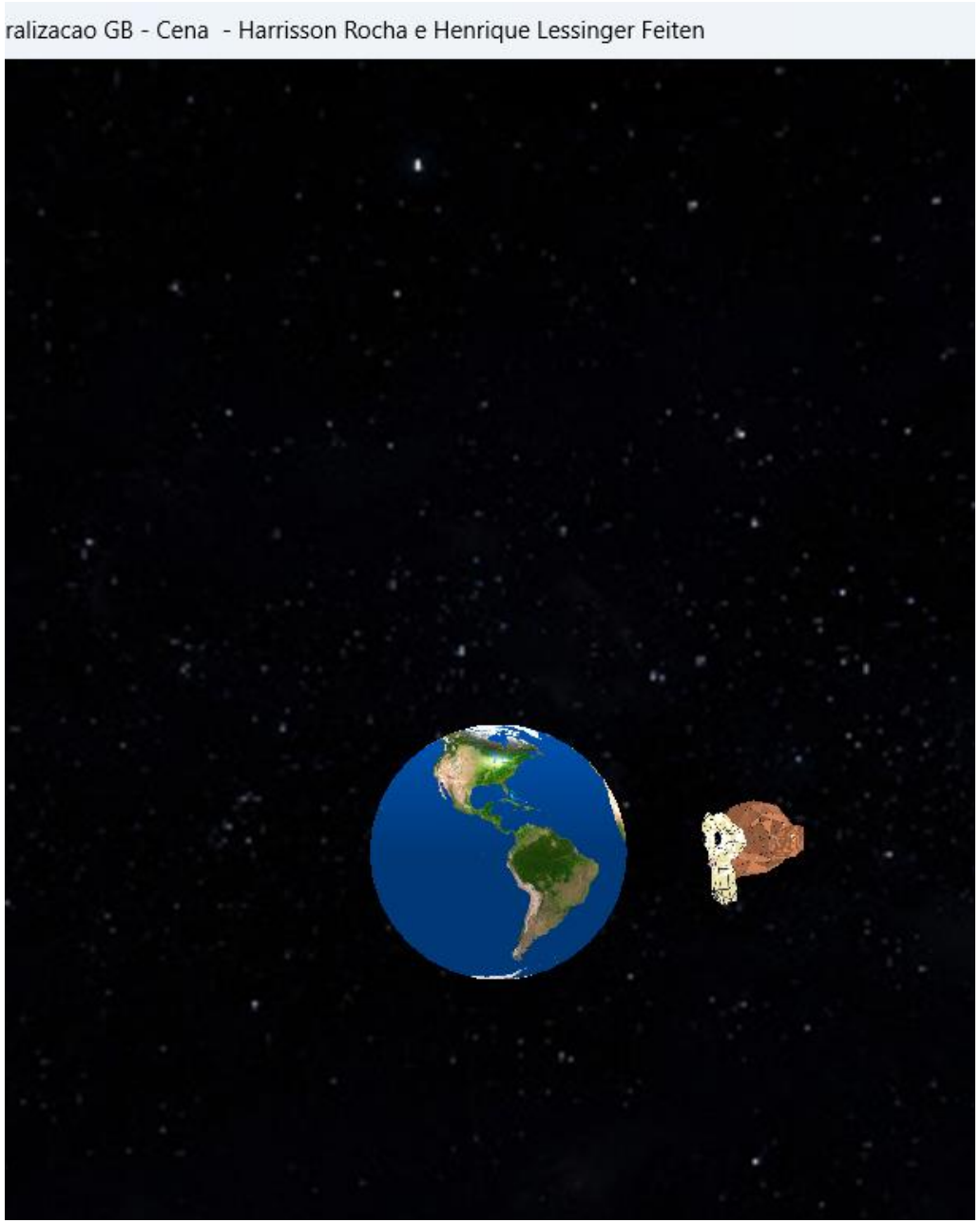
Leitor / Visualizador de Cenas 3D com OpenGL

Atividade Acadêmica de Computação Gráfica

Alunos: Harrisson Rocha e Henrique Feiten

UNISINOS

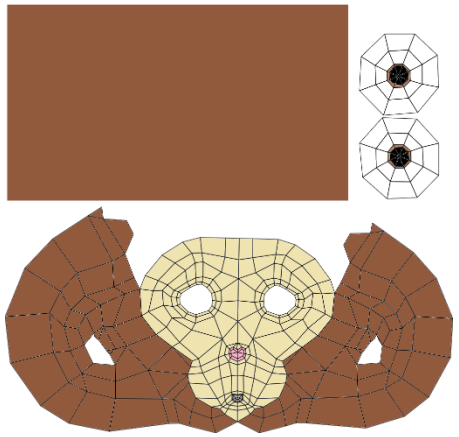
DESAFIE O AMANHÃ.

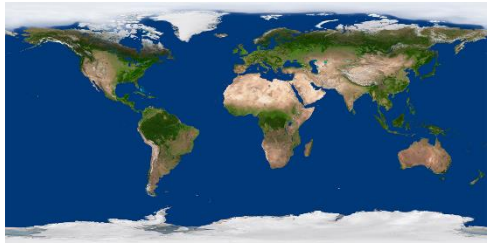


Características do projeto

- Número de objetos carregados: 3
- Composição da cena: Objetos disponibilizados no material (sem interferência de softwares de modelagem)
- Iluminação de Phong
- Curva: Bezier

UNISINOS | 2024





Parâmetros técnicos

- 3D models:
  - Earth.obj
  - Bola.obj
  - SuzanneTriTextured.obj
- Arquivos de material:
  - Earth.mtl
  - Bola.mtl
  - SuzanneTriTextured.mtl

```
//Paths dos arquivos
string tresdModelsPaths = "../../3D_Models";
string backgroundPath = tresdModelsPaths + "/Background/bola.obj";
string earthPath = tresdModelsPaths + "/Earth/bola.obj";
string suzannePath = tresdModelsPaths + "/Suzanne/SuzanneTriTextured.obj";
```

Parâmetros técnicos

- Classes adicionais:
  - Geração da curva:
    - Bezier
  - Camera
  - Mesh

- Exemplo das chamadas:

```
//Background
loadOBJ(backgroundPath);
loadMTL(tresdModelsPaths + "/Background/" + mtlFile);
GLuint textureIDBackground = loadTexture("../../3D_Models/Background/Stars_texture.jpg");
GLuint VAOBackground = setupGeometry();
Mesh background;
background.initialize(VAOBackground, positions.size() / 3, &shader, textureIDBackground, glm::vec3(0.0f, 0.0f, 0.0f), glm::vec3(50.0f));
clearVertexForNext();
```

```
std::vector<glm::vec3> controlPoints = createCircularCurve(radius, numPoints, suzanneSpeed);

Bezier bezier;
bezier.setControlPoints(controlPoints);
bezier.setShader(&shader);
bezier.generateCurve(100);
int nbCurvePoints = bezier.getNbCurvePoints();
int i = 0;
```

```
camera.update();
background.update();
background.draw();

earth.updatePosition(glm::vec3(earthX, earthY, 0.0f));
earth.update();
```

Parâmetros técnicos

```glsl
#version 450 core

in vec3 finalColor;
in vec3 scaledNormal;
in vec3 fragPos;
in vec2 texCoord;

//Propriedades do material do objeto
uniform vec3 ka;
uniform float kd;
uniform vec3 ks;
uniform float q;

//Propriedades da fonte de luz
uniform vec3 lightPos;
uniform vec3 lightColor;

//Posição da câmera
uniform vec3 cameraPos;

//Buffer de saída (color buffer)
out vec4 color;

//buffer de textura
uniform sampler2D colorBuffer;

void main()
{
    // Ambient
    vec3 ambient =  lightColor * ka;
    // Diffuse
    vec3 N = normalize(scaledNormal);
    vec3 L = normalize(lightPos - fragPos);
    float diff = max(dot(N, L),0.0);
    vec3 diffuse = diff * lightColor * kd;

    // Specular
    vec3 R = reflect(-L,N);
    vec3 V = normalize(cameraPos - fragPos);
    float spec = pow(max(dot(R,V),0.0),q);
    vec3 specular = spec * ks * lightColor;

    vec4 texColor = texture(colorBuffer,texCoord);
    vec3 result = (ambient + diffuse) * vec3(texColor) + specular;

    color = vec4(result, 1.0f);
}
```

- Fragment Shader:
  - Cálculo da cor final dos fragmentos
  - Iluminação
  - Textura
- Vertex Shader:
  - Posições dos vértices no espaço
  - Calculo da posição dos fragmentos
  - Ajuste de coordenadas
  - Joga a normal do vértice para o FS

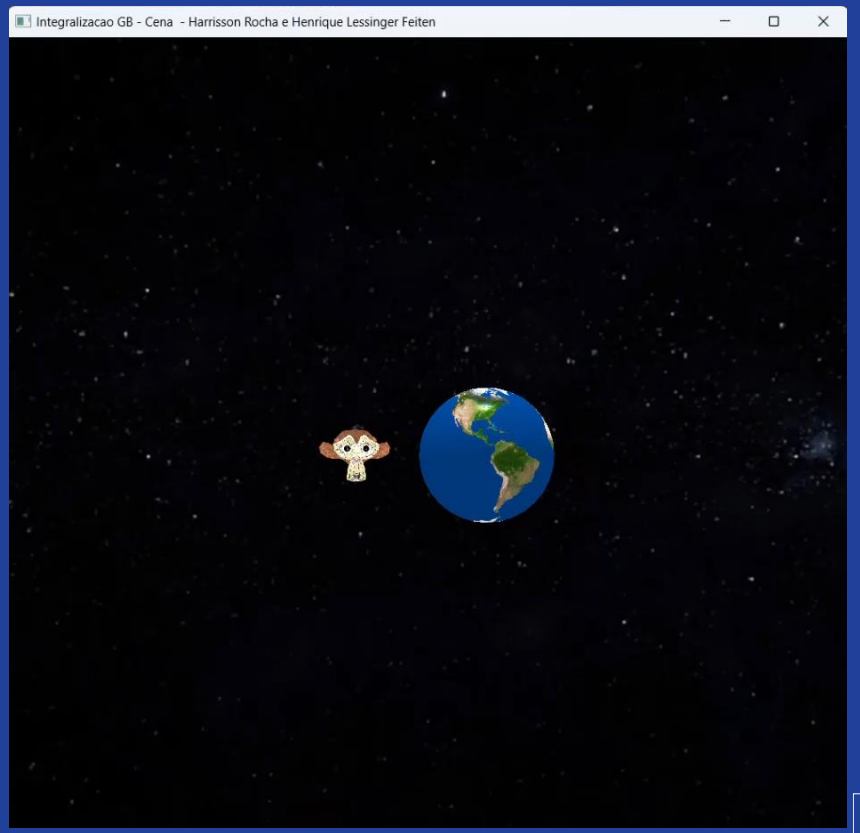
```
camera.initialize(&shader, width, height, 0.05f, 0.05f, -90.0f, glm::vec3(0.0f, 0.0f, -1.0f), glm::vec3(1.5f, 0.0f, 15.0f));

shader.setVec3("ka", ka[0], ka[1], ka[2]);
shader.setFloat("kd", 0.5);
shader.setVec3("ks", ks[0], ks[1], ks[2]);
shader.setFloat("q", ns);
shader.setVec3("lightPos", -2.0f, 100.0f, 2.0f);
shader.setVec3("lightColor", 1.0f, 1.0f, 1.0f);
```

Funções main

- Key_callback
- mouse_callback
- setupGeometry
- loadOBJ
- loadMTL
- clearVertexForNext
- generateControlPointsSet
- createCircularCurve

- Controle do objeto
- Controle de câmera
- Geração dos VAOs
- Leitura dos objetos
- Leitura de materiais
- Limpeza dos vetores globais para nova leitura de objeto
- Criação de conjunto de pontos de controle
- Geração de pontos na curva tridimensional

Exemplo Execução



UNISINOS | 2024

UNISINOS

DESAFIE
O AMANHÃ.

Obrigado!