Sistema Solar 2D/3D

Relatório de Projecto da Unidade Curricular
Computação Gráfica

2 de Junho de 2019

Departamento de
Informática

Daniel Valente nº35453
Carolina Lopes nº38338
Cristina Pinto nº38991

Sinopse

O presente documento encerra o relatório de projeto da unidade curricular de *Computação Gráfica*. Neste projeto foi abordado vários temas lecionados nas aulas de Computação Gráfica que englobam desde iluminação, a projeção e por fim texturização.

O conhecimento teórico e pratico adquirido nesta unidade curricular foi utilizado para a renderização de um sistema solar á escala com texturas e velocidades reais

Daniel Valente nº35453
Carolina Lopes nº38338
Cristina Pinto nº38991
Covilhã, Portugal
2 de Junho de 2019

Conteúdo

Capítulo 1

Motivação

O presente relatório diz respeito ao projeto final da cadeira de Computação Gráfica, onde foi desenvolvido um sistema solar em C++ com auxilio de Opengl. Neste relatório será exposto a motivação para a realização do projeto, as Tecnologias Utilizadas, a várias etapas do desenvolvimento, iremos descrever o Software desenvolvido pelo grupo, trabalhos futuros e considerações finais.

A realização deste projeto é motivada pelo desejo de aplicar os conhecimentos aprendidos em aula e aprofundar um pouco mais os mesmos, com este projeto para além de pôr em pratica os conhecimentos desta unidade curricular, também expandimos os nosso conhecimento sobre o sistema solar.

Capítulo 2

Tecnologias Utilizadas

Neste projeto as tecnologias utilizadas foram:

- **VisualStudio** utilizado como o IDE de programação para este projeto. É um ambiente de desenvolvimento integrado (IDE) da Microsoft para desenvolvimento de software especialmente dedicado ao .NET Framework e às linguagens Visual Basic (VB), C, C++, C# (C Sharp) e F# (F Sharp).
- **OpenGl** (Open Graphics Library) é uma API livre utilizada na computação gráfica, para desenvolvimento de aplicações gráficas, ambientes 3D, jogos, entre outros. É uma API (Application Programming Interface), termo usado para classificar uma biblioteca de funções específicas disponibilizadas para a criação e desenvolvimento de aplicativos em determinadas linguagens de programação. A OpenGL foi produzida com C e C++.
- **Glew** (OpenGL Extension Wrangler Library) é uma biblioteca C / C ++ multi-plataforma que ajuda na consulta e no carregamento de extensões OpenGL. O GLEW fornece mecanismos de tempo de execução eficientes para determinar quais as extensões OpenGL suportadas na plataforma de destino.
- **GLFW** (Graphics Library Framework) é uma biblioteca para uso com OpenGL. Permite que programadores possam criar e gerir janelas e contextos OpenGL, assim como interagir com joysticks, mouses e teclados.
- **Blender** é um programa de computador de código aberto, desenvolvido pela Blender Fundation, para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D, tais como jogos, apresentações e outros.

Capítulo 3

Etapas de Desenvolvimento

O desenvolvimento deste projeto passou por várias fases, de acordo com os conteúdos abordados nas aulas teóricas e práticas da unidade curricular.

Assim, numa primeira fase, procedeu-se à análise dos objetivos do trabalho e do código inicialmente fornecido.

De seguida, foram feitos ajustamentos ao código, nomeadamente:

- **Modelação do sol, planetas e respetivos satélites e desenho dos mesmos** - O desenho dos planetas foi feito em blender, através do qual se projetaram as esferas correspondentes a cada planeta. (Carolina Lopes)

- **Aplicação de Iluminação**, através de foco de luz proveniente do sol (Carolina Lopes)

- **Aplicação de Texturas / Texturização** ao sol e planetas para aumentar o realismo - A texturização foi feita com recurso ao glslcookbook [4] e com utilização de texturas retiradas de [2] (Daniel Valente)

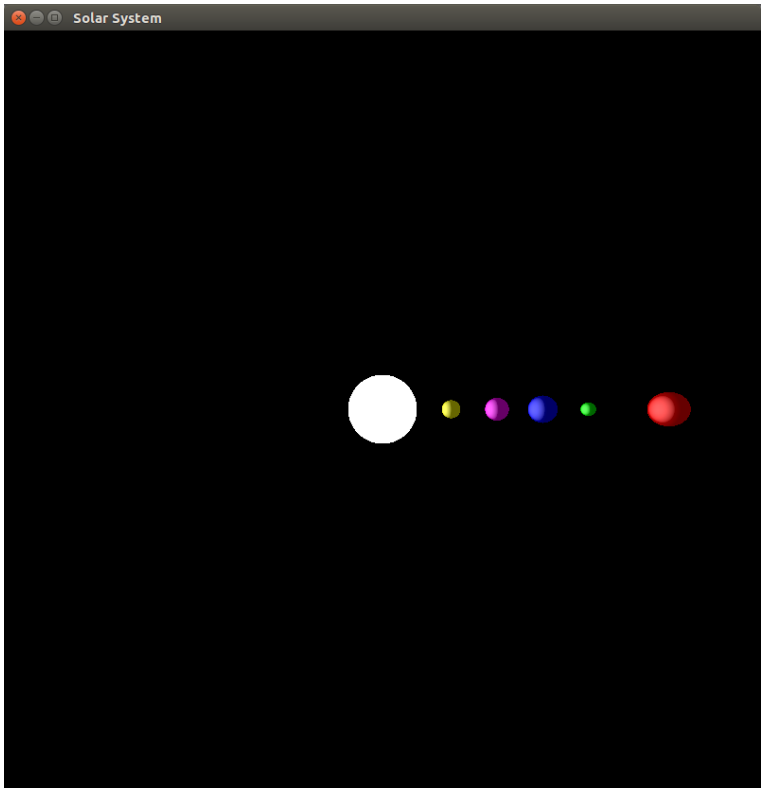
- **Definição de movimentos de rotação e translação** (Cristina Pinto)

- **Movimentação da câmara** - A câmara pode mover-se para qualquer posição do sistema solar, sendo que este movimento afeta a luz que incide sobre os elementos do sistema solar. (Daniel Valente)

- **Elaboração de menus** - Menus que permitam obter informação sobre os elementos do sistema solar (por exemplo, planetas). (Carolina Lopes)

- **Renderização**

- **Realização relatório** (Cristina Pinto)



Fase Inicial [3]

Fase Final [3]

Capítulo 4

Descrição do funcionamento do Software

- **Carregamento de objetos espaciais** - Cada objeto é carregado para o programa com ajuda da função *ObjMesh::load* do CookBook [4]. Os objetos foram desenvolvidos em **Blender** e exportados de modo a serem aplicados neste projeto. O carregamento dos Objetos encontra-se implementado na inicialização da classe SceneMultiLight pelo que apenas são carregados uma vez para a memoria, evitando desperdícios.

Planeta	Distância em relação ao Sol
Mercúrio	57.9
Vénus	108.2
Terra	149.6
Marte	227.9
Júpiter	778.3
Saturno	1427.0
Úrano	2871.0
Neptuno	4497.0

Tabela 4.1: Distância em relação ao Sol [1]

- **Definição de Movimentos** - Os planetas orbitam em torno do sol, na rota definida e de acordo com a sua velocidade. A função que determina a rotação dos planetas é a função *glm::rotate* que recebe como parâmetros o modelo, a velocidade de rotação do respetivo planeta e o eixo em que se efetua essa rotação. A translação de cada planeta em torno do sol é efetuada através de um *glm::translate* que leva como parâmetros uma matriz de 4x4 e um *glm::vec3* onde serão indicados o x, o y e o z. A posição do x é dada por sin(tempAngle)*distancePlante, a posição de y é sempre 0.0f porque não queremos movimento nesse eixo e a posição nos eixos de z é dada por

cos(tempAnlge)*distancePlanet.Note se que o valor denominado de temAngle é dado pela expressão (anglePlanet*velocityPlanet)/180*3.14159.

```
EarthAngle += EarthSpeed;
while (EarthAngle > 360.0)
    EarthAngle -= 360.0;
float temp3Angle = (EarthAngle / 180.0) * 3.14159;
model = glm::translate(glm::mat4(1.f), glm::vec3(sin(temp3Angle) * 149.6f, 0.f, cos(temp3Angle) * 149.6f));
model = glm::rotate(model, glm::radians((GLfloat)glfwGetTime() * 20.0f), glm::vec3(0.0f, 1.0f, 0.0f));
model = glm::scale(model, glm::vec3(0.003f, 0.003f, 0.003f));
textureLoad(texEart, sEart, tEart);
setMatrices();
earth->render();
```

- **Texturização** - As texturas dos diferentes planetas foram obtidas no site da NASA [2] e definidas no na função *Texture::loadPixels()*, utilizada uma vez ao iniciar o programa e *textureLoad()* para renderização.

```
unsigned char *Texture::loadPixels(const std::string &fName, int & width, int & height) {
    int bytesPerPix;
    stbi_set_flip_vertically_on_load(true);
    unsigned char *data = stbi_load(fName.c_str(), &width, &height, &bytesPerPix, 4);
    return data;
}
```

```
SceneMultiLight::SceneMultiLight()
{

    sun = ObjMesh::load("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\objects\\Sun\\Sun.obj", true);
    texSun = Texture::loadPixels("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\Texturas\\2k_sun.jpg", sSun, tSun);


    mercury = ObjMesh::load("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\objects\\Mercury\\Mercury.obj", true);
    texMerc = Texture::loadPixels("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\Texturas\\2k_mercury.jpg", sMerc, tMerc);


    venus = ObjMesh::load("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\objects\\Venus\\Venus.obj", true);
    texVenu = Texture::loadPixels("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\Texturas\\2k_venus.jpg", sVenu, tVenu);


    earth = ObjMesh::load("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\objects\\Earth\\Earth.obj", true);
    texEart = Texture::loadPixels("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\Texturas\\2k_earth_daymap.jpg", sEart, tEart);


    mars = ObjMesh::load("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\objects\\Mars\\Mars.obj", true);
    texMars = Texture::loadPixels("C:\\Users\\carol\\Desktop\\Projeto\\Projeto\\Texturas\\2k_mars.jpg", sMars, tMars);
```

```
void SceneMultiLight::render()
{
    view = glm::lookAt(vec3(eyex, 50.f, eyez), vec3(camx, 0.0f, camz), vec3(upx, 1.0f, upz));

    if (SceneMultiLight::paused == false) {

        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        glClearColor(0.f, 0.f, 0.f, 0.f);

        prog.setUniform("Light.Position", glm::vec4(0.0f, 0.0f, 0.0f, 1.0f));
        prog.setUniform("Material.Kd", 1.f, 1.f, 1.f);
        prog.setUniform("Material.Ks", 1.f, 1.f, 1.f);
        prog.setUniform("Material.Ka", 0.1f, 0.1f, 0.1f);
        prog.setUniform("Material.Shininess", 100.0f);

        model = mat4(1.0f);
        model = glm::translate(model, vec3(.0f, .0f, 0.0f));
        model = glm::scale(glm::mat4(1.f), glm::vec3(0.1f, 0.1f, 0.1f));
        textureLoad(texSun, sSun, tSun);
        setMatrices();
        sun->render();

        MercuryAngle += MercurySpeed;
        while (MercuryAngle > 360.0)
            MercuryAngle -= 360.0;
        float temp1Angle = (MercuryAngle / 180.0) * 3.14159;
        model = glm::translate(glm::mat4(1.f), glm::vec3(sin(temp1Angle) * 57.9f, 0.f, cos(temp1Angle) * 57.9f));
        model = glm::rotate(model, glm::radians((GLfloat)glfwGetTime() * 20.0f), glm::vec3(0.0f, 1.0f, 0.0f));
        model = glm::scale(model, glm::vec3(0.01f, 0.01f, 0.01f));
        textureLoad(texMerc, sMerc, tMerc);
        setMatrices();
        mercury->render();
```

- **Iluminação** - A cena é iluminada por uma fonte de luz de um ponto único localizado no centro da estrela (Sol). A Estrela/sol é iluminado por uma luz ambiente brilhante. O modelo de iluminação usado é o modelo de Phong, definido na função *phongModel()*, no shader.

```
void phongModel( vec3 pos, vec3 norm, out vec3 ambAndDiff, out vec3 spec ) {
    vec3 s = normalize(vec3(Light.Position) - pos);
    vec3 v = normalize(-pos.xyz);
    vec3 r = reflect( -s, norm );
    vec3 ambient = Light.Intensity * Material.Ka;
    float sDotN = max( dot(s,norm), 0.0 );
    vec3 diffuse = Light.Intensity * Material.Kd * sDotN;
    spec = vec3(0.0);
    if( sDotN > 0.0 )
        spec = Light.Intensity * Material.Ks *
                pow( max( dot(r,v), 0.0 ), Material.Shininess );
    ambAndDiff = ambient + diffuse;
}

void main() {
    vec3 ambAndDiff, spec;
    vec4 texColor = texture( Tex1, TexCoord );
    phongModel( Position, Normal, ambAndDiff, spec );
    FragColor = (vec4( ambAndDiff, 1.0 ) * texColor) + vec4(spec,1.0);
}
```

- **Zooming e Pausa** - As funcionalidades de zooming e pausa foram implementadas através da função *SceneMultiLight::keyfunc()*. Assim, a barra de espaço pode ser usada para parar ou retomar a animação. As teclas "UP", "DOWN", "Rigth"e "LEFT"servem para navegar e aproximar dos vários planetas.

```
if (glfwGetKey(window, GLFW_KEY_SPACE) == GLFW_PRESS) {
    SceneMultiLight::paused = !SceneMultiLight::paused;

}

else if (key == GLFW_KEY_DOWN) {
    if(eyex<=1500.0f)
        eyex += 20.f;
}

else if (key == GLFW_KEY_UP) {
    if(eyex>=0.0f)
        eyex -= 20.f;
}

else if (key == GLFW_KEY_LEFT) {
    if(eyez<=150.0f)
        eyez += 20.f;
}

else if (key == GLFW_KEY_RIGHT)
{
    if(eyez<=-150.0f)
        eyez -= 20.f;
}
```

- **Menus** - Possibilidade de ler informação sobre planetas. Para aceder á informação sobre os planetas basta clicar nos números de 0 a 8(ordem crescente a partir do sol), cada numero corresponde a um planeta e apresenta a respetiva informação desse planeta. Esta função também está presente na *SceneMultiLight::keyfunc()*.

Capítulo 5

Trabalhos Futuros

Neste projecto implementar um sistema solar interactivo através da utilização da linguagem C++, bem como as bibliotecas OpenGL, GLFW, GLEW e GLM.

No futuro, este projeto poderá continuar a ser desenvolvido em duas vertentes, uma mais macro e outra micro.

Numa vertente mais macro, uma aplicação desta experiência interativa a toda a Via Láctea, ou até outras galáxias, permitindo a exploração de outros objetos celestes, nomeadamente estrelas e buracos negros.

Na perspetiva mais micro, o desevolvimento deste projeto poderia fazer uma maior aproximação aos vários planetas, nomeadamente a Terra, explorando os vários elementos que constituem cada planeta em particular.

Após a exploração das vertentes macro e micro da galáxia e planetas, poderá ainda ser desenvolvido um jogo lúdico-pedagógico interativo para utilização por crianças ou escolas.

Capítulo 6

Considerações Finais

O desenvolvimento deste projeto ajudou no aprofundamento de competências de Computação Gráfica, adquiridas nas aulas teóricas e práticas, nomedamente na linguagem de programação C++, bem como das tecnologias OpenGL, Glew, GLFW.

Este trabalho permitiu ainda aos elementos do grupo ter um maior conhecimento de como elaborar uma aplicação interativa, com modelação de objetos, aplicação de iluminação e texturas, definição de movimentos de rotação e translação e movimentação da câmara dentro do cenário criado.

Por outro lado, permitiu o grupo aprofundasse os conhecimentos sobre os elementos constituintes do Sistema Solar.

Bibliografia

[1] Planetary fact sheet - metric. `https://nssdc.gsfc.nasa.gov/planetary/factsheet`.

[2] Solar system scope. `https://www.solarsystemscope.com/textures/`.

[3] Wallace Rao. Simulation of solar system based on opengl.

[4] D. Wolff. *OpenGL 4 Shading Language Cookbook, Second Edition*. Packt Publishing, 2013.