Breno Pinheiro Gallo de Sá – 2110183

Felippe Petrasso Fonseca Hübner – 2110870

Projeto 1 – Computação gráfica

Escolha do tema:

-Optamos pela escolha da opção 1, a simulação do mini sistema solar, utilizando texturas para exibir o espaço, a lua, o sol e Vênus, e utilizando mapeamento de um textura maior para a Terra simulando o movimento de rotação em seu eixo.

Execução do projeto:

-Foi fornecido pelo professor o modelo do grafo de cena e todos os arquivos shaders necessários para a execução, portanto nãofoi necessário alterar grande partes dos arquivos dados, os principais arquivos que foram alterados foram: a classe disk.py foi criada para poder renderizar os astros como discos na tela, animation.py, para executar a animação do sistema, incluindo o movimento de translação dos planetas em torno do sol, da lua ao redor da Terra e também a rotação da Terra em seu próprio eixo, e por fim a main\_2d.py, controlando a renderização da cena como um todo e definição das texturas.

-A definição dos shaders e texturas foi feita de forma separada para os 3 tipos de astros definidos como, astros estáticos, sendo a lua e vênus, que apenas têm suas texturas impressas no disco, shader com textura móvel, para a Terra em específico por conta de sua rotação, por último shader emissivo, para o Sol, fazendo-o brilhar. A seguir está uma parte do código responsável por essas definições:

*# --- ARQUIVOS DE SHADER ---*

    vertex\_shader\_file = os.path.join(ilum\_vert\_path, 'vertex\_texture.glsl')

    fragment\_shader\_file = os.path.join(ilum\_vert\_path, 'fragment\_texture.glsl')

*# --- SHADER PARA PLANETAS ESTÁTICOS (VÊNUS, LUA) ---*

    static\_planet\_shader = Shader()

    static\_planet\_shader.AttachVertexShader(vertex\_shader\_file)

    static\_planet\_shader.AttachFragmentShader(fragment\_shader\_file)

    static\_planet\_shader.Link()

    static\_planet\_shader.UseProgram()

    static\_planet\_shader.SetUniform("lpos", glm.vec4(0.0, 0.0, 4.0, 1.0))

    static\_planet\_shader.SetUniform("lamb", glm.vec4(0.2, 0.2, 0.2, 1.0))

    static\_planet\_shader.SetUniform("ldif", glm.vec4(1.0, 1.0, 1.0, 1.0))

    static\_planet\_shader.SetUniform("lspe", glm.vec4(1.0, 1.0, 1.0, 1.0))

    static\_planet\_shader.SetUniform("mamb", glm.vec4(1.0, 1.0, 1.0, 1.0))

    static\_planet\_shader.SetUniform("mdif", glm.vec4(0.8, 0.8, 0.8, 1.0))

    static\_planet\_shader.SetUniform("mspe", glm.vec4(0.5, 0.5, 0.5, 1.0))

    static\_planet\_shader.SetUniform("mshi", 32.0)

    static\_planet\_shader.SetUniform("u\_texture\_offset", 0.0)

*# --- SHADER PARA A TERRA (COM TEXTURA MÓVEL) ---*

    global earth\_shader

    earth\_shader = Shader()

    earth\_shader.AttachVertexShader(vertex\_shader\_file)

    earth\_shader.AttachFragmentShader(fragment\_shader\_file)

    earth\_shader.Link()

    earth\_shader.UseProgram()

    earth\_shader.SetUniform("lpos", glm.vec4(0.0, 0.0, 4.0, 1.0))

    earth\_shader.SetUniform("lamb", glm.vec4(0.2, 0.2, 0.2, 1.0))

    earth\_shader.SetUniform("ldif", glm.vec4(1.0, 1.0, 1.0, 1.0))

    earth\_shader.SetUniform("lspe", glm.vec4(1.0, 1.0, 1.0, 1.0))

    earth\_shader.SetUniform("mamb", glm.vec4(1.0, 1.0, 1.0, 1.0))

    earth\_shader.SetUniform("mdif", glm.vec4(0.8, 0.8, 0.8, 1.0))

    earth\_shader.SetUniform("mspe", glm.vec4(0.5, 0.5, 0.5, 1.0))

    earth\_shader.SetUniform("mshi", 32.0)

    earth\_shader.SetUniform("u\_texture\_offset", 0.0)

*# --- SHADER PARA O SOL (EMISSIVO) ---*

    sun\_shader = Shader()

    sun\_shader.AttachVertexShader(vertex\_shader\_file)

    sun\_shader.AttachFragmentShader(fragment\_shader\_file)

    sun\_shader.Link()

    sun\_shader.UseProgram()

    sun\_shader.SetUniform("lpos", glm.vec4(0.0, 0.0, 4.0, 1.0))

    sun\_shader.SetUniform("lamb", glm.vec4(0.2, 0.2, 0.2, 1.0))

    sun\_shader.SetUniform("ldif", glm.vec4(1.0, 1.0, 1.0, 1.0))

    sun\_shader.SetUniform("lspe", glm.vec4(1.0, 1.0, 1.0, 1.0))

    sun\_shader.SetUniform("mamb", glm.vec4(5.0, 5.0, 5.0, 1.0))

    sun\_shader.SetUniform("mdif", glm.vec4(0.0, 0.0, 0.0, 1.0))

    sun\_shader.SetUniform("mspe", glm.vec4(0.0, 0.0, 0.0, 1.0))

    sun\_shader.SetUniform("mshi", 1.0)

    sun\_shader.SetUniform("u\_texture\_offset", 0.0)

    glUseProgram(0)

*# --- CARREGAMENTO DAS TEXTURAS ---*

    sun\_texture = Texture("decal", sun\_image\_path)

    earth\_texture = Texture("decal", earth\_image\_path)

    moon\_texture = Texture("decal", moon\_image\_path)

    venus\_texture = Texture("decal", venus\_image\_path)

    space\_texture = Texture("decal", space\_image\_path)  *# <-- NOVO*

-As animações são controladas no arquivo animation.py, pela função SolarSystemAnimation representada a baixo:

class SolarSystemAnimation(Engine):

    def \_\_init\_\_(self, earth\_orbit\_transform, moon\_orbit\_transform, earth\_spin\_transform, sun\_spin\_transform, venus\_orbit\_transform, venus\_spin\_transform):

*self*.earth\_orbit\_transform = earth\_orbit\_transform

*self*.moon\_orbit\_transform = moon\_orbit\_transform

*self*.earth\_spin\_transform = earth\_spin\_transform

*self*.sun\_spin\_transform = sun\_spin\_transform

*self*.venus\_orbit\_transform = venus\_orbit\_transform

*self*.venus\_spin\_transform = venus\_spin\_transform

    def Update(self, time):

*self*.earth\_orbit\_transform.LoadIdentity()

*self*.moon\_orbit\_transform.LoadIdentity()

*self*.earth\_spin\_transform.LoadIdentity()

*self*.sun\_spin\_transform.LoadIdentity()

*self*.venus\_orbit\_transform.LoadIdentity()

*self*.venus\_spin\_transform.LoadIdentity()

        earth\_orbit\_speed = 0.5

        moon\_orbit\_speed = 3.0

        earth\_spin\_speed = 2.0

        sun\_spin\_speed = 0.2

        venus\_orbit\_speed = -1.5

        venus\_spin\_speed = 2.0

        earth\_orbit\_angle\_rad = time \* earth\_orbit\_speed

        moon\_orbit\_angle\_rad = time \* moon\_orbit\_speed

        earth\_spin\_angle\_rad = time \* earth\_spin\_speed

        sun\_spin\_angle\_rad = time \* sun\_spin\_speed

        venus\_orbit\_angle\_rad = time \* venus\_orbit\_speed

        venus\_spin\_angle\_rad = time \* venus\_spin\_speed

*self*.earth\_orbit\_transform.Rotate(math.degrees(earth\_orbit\_angle\_rad), 0, 0, 1)

*self*.moon\_orbit\_transform.Rotate(math.degrees(moon\_orbit\_angle\_rad), 0, 0, 1)

*self*.earth\_spin\_transform.Rotate(math.degrees(earth\_spin\_angle\_rad), 0, 0, 1)

*self*.sun\_spin\_transform.Rotate(math.degrees(sun\_spin\_angle\_rad), 0, 0, 1)

*self*.venus\_orbit\_transform.Rotate(math.degrees(venus\_orbit\_angle\_rad), 0, 0, 1)

*self*.venus\_spin\_transform.Rotate(math.degrees(venus\_spin\_angle\_rad), 0, 0, 1)

-Por fim, para o desenho da cena na própria main há a função update\_and\_draw() que é responsável por desenhar a cena e ficar atualizando-a com as novas posições dos planetas e a movimentação da textura da Terra:

def update\_and\_draw():

    global last\_time, scene, state, earth\_disk

    current\_time = glfw.get\_time()

    last\_time = current\_time

    glClearColor(0.0, 0.0, 0.1, 1.0)

    glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

*# --- ATUALIZAÇÃO DO DESLOCAMENTO DA TEXTURA DA TERRA ---*

    earth\_texture\_rotation\_speed = 0.5  *# Velocidade da rotação da textura*

    texture\_offset = current\_time \* earth\_texture\_rotation\_speed

    if earth\_disk is not None:

        earth\_disk.update\_texcoords(texture\_offset)

*# --- DESENHO DA CENA ---*

    if scene and state:

        for e in scene.engines:

            e.Update(current\_time)

        scene.Render(state.camera)