Alunos: Silvana Trindade e Maurício André Cinelli

Sistema Solar

Descrição: *Faça um modelo do nosso sistema solar com uma nave que possa navegar neste sistema.*

Este trabalho possui os seguintes planetas: Mercúrio, Vênus, Terra (e lua), Marte, Júpiter, Saturno, Urano e Netuno. As órbitas são elípticas, com o sol em uma das extremidades. O tamanho dos planetas, as distâncias entre eles e as velocidades que cada planeta orbita o sol são proporcionais às de [1]. Além disso, para melhor visualização, foi feito o trajeto de cada órbita com cubos, visto que ao fazermos os tamanhos dos planetas de forma proporcional à realidade, alguns ficaram pequenos. Para simular a estrutura física dos planetas, utilizamos imagens JPG e PNG como texturas. Para a leitura dos arquivos de textura, utilizamos a biblioteca STB Image [2].

Existem duas câmeras: a modo Deus, e a câmera da nave. A câmera modo Deus é permitido apenas o zoom utilizando a rolagem do mouse. A câmera da nave permite apenas a navegação utilizando o teclado, movimentando uma nave de fato ao mesmo tempo, podendo ser vista pela câmera modo Deus. Fizemos um *skybox*, ou seja, um espaço estrelado ao redor do nosso sistema solar, a fim de dar mais realidade ao mesmo e limitamos o movimento de ambas as câmeras para impedir que o usuário "saia" do skybox.

Para a nave, desenhamos-a utilizando o software Blender, e exportando a mesma em formato "OBJ" para a leitura no programa. O arquivo OBJ contém todos os vértices, normais e faces dos objetos exportados, além de um arquivo de material na qual cada objeto utiliza. Cada objeto tem suas características agrupadas(vértices, normais e faces), junto com a referência do material de cada objeto. Com os objetos lidos, basta percorrer e desenhar as faces. A "cabine" da nave foi utilizado transparência para simular o efeito de vidro. Uma descrição e especificação do formato pode ser vista em [3].

Instruções de Uso

Para alternar entre a câmera modo deus e modo nave, aperte a tecla [c].

Comandos modo Deus

No modo Deus, apenas zoom pelo mouse funciona. Portanto, com o mouse, posicione-o sobre o ponto que deseja dar zoom (ou tirar zoom), e role o botão do meio do mouse para frente para aumentar o zoom, e para trás para diminuí-lo.

Comandos modo nave

Para mover a nave para frente, pressione a seta para cima
Para mover a nave para a esquerda, pressione a seta para esquerda
Para mover a nave para a direita, pressione a seta para direita

Não é possível mover para trás.

Para rotacionar a nave para a esquerda, pressione a tecla [a].
Para rotacionar a nave para a direita, pressione a tecla [d].
Para rotacionar a nave para cima, pressione a tecla [w].
Para rotacionar a nave para baixo, pressione a tecla [s].

Note que o CAPS LOCK deve estar **desativado**.

Descrição das Classes

Vector3

A Classe *Vector3* foi baseada no código da classe apresentada no Capítulo 6 da referência [2]. É uma classe com o intuito de facilitar nos cálculos de vetores e pontos, sem ter que utilizar uma biblioteca mais completa. A classe contempla operações básicas entre vetores, como por exemplo a soma e subtração, e de vetores com escalares, como multiplicação e divisão. Faz também o cálculo da magnitude, normal, produto interno e externo.

Anel

A classe Anel é utilizada para desenhar e gerenciar os anéis de Saturno. Os anéis de Saturno foram feitos utilizando um torus achatado. Nesta classe, é feito o controle do material, calcula a rotação, órbita, e carrega e atribui uma textura ao objeto. Passamos a mesma órbita de saturno para cá, de modo que ele faça a translação de maneira síncrona.

As órbitas (aqui e na classe Planeta) são calculadas para formar uma elipse, através da fórmula [4]:

$$x = deslocamento + raioOrbita \times cos(angulo \div 180 \times \pi)$$
$$y = deslocamento + raioOrbita \times sin(angulo \div 180 \times \pi)$$

Com estes valores, é realizado uma translação após a rotação do objeto.

Para impedir que o movimento de um objeto afete o outro utilizamos o par *glPushMatrix* e *glPopMatrix* ao desenhar cada um.

Camera

A classe Camera gerencia os movimentos das câmeras, bem como as limitações delas.
Para o modo deus, é feito apenas o controle de zoom, onde recebe o ponto onde o usuário está com o mouse encima, para realizar o zoom em direção ao mesmo.

Para a câmera da nave, são implementados métodos para movimentar para os lados, e pra frente, utilizando cálculo de vetores, mantendo sempre o vetor de direção (no caso de rotação). Resumindo a ideia de como são feitos os movimentos:

- Movimento para frente: Soma com vetor de direção

- Movimento para esquerda: Soma com vetor perpendicular, achado pelo produto externo da direção com o vetor *up*
- Movimento para esquerda: Soma com o inverso do vetor perpendicular do movimento à esquerda
- Rotação para esquerda: rotaciona o vetor de direção ao redor do vetor *up* em 1º por vez
- Rotação para direita: rotaciona o vetor de direção ao redor do vetor *up* em -1º por vez
- Rotação para cima: rotaciona em -1º o vetor de direção sobre um vetor perpendicular encontrado pelo produto externo entre o vetor up e dir
- Rotação para baixo: rotaciona em 1º o vetor de direção sobre um vetor perpendicular encontrado pelo produto externo entre o vetor up e dir

A câmera é projetada utilizando a função *gluLookAt*.

Galaxia

A galáxia é feita com uma esfera gigante, coberta por uma textura de um céu estrelado.
É muito similar à classe Planeta, mas não tem nenhum cálculo de rotação e órbita. É apenas o desenho da esfera com textura.

Material

A classe Material é responsável por armazenar características de um material lido de um arquivo OBJ. Cada material possui nome, transparência, coeficiente especular, e cores ambiente, difusa e especular.

Além disso a classe possui um método chamado *aplica*, responsável por chamar funções do openGL para aplicar o material ao próximo objeto a ser desenhado.

Nave

Responsável por ler arquivo OBJ com as informações dave, e os materiais associados à cada objeto. Estes dados são armazenados, para que na função *desenha* sejam usados para renderizar cada triângulo dos objetos.

A função desenha, desenha a nave na posição da câmera da nave, e com a mesma rotação. Percorre os objetos lidos do arquivo, aplicando o material associado a cada um, e desenha os triângulos na ordem correta, montando as faces do objeto.

Se o objeto for transparente, desabilitamos o teste de profundidade, dando o efeito de vidro.

Na função *carregaObjetos*, cada linha do arquivo OBJ é percorrido, identificando cada característica, armazenando-as nas variáveis em cada sub-objeto.
No arquivo há o nome do arquivo de material a ser lido. Quando esta linha aparece, os materiais são lidos, assim quando os objetos são lidos, a referência do material pode ser pega.

O detalhe é que as faces são indicadas por indices de vértices e normais, indicando os vértices que formam cada triângulo.

A leitura é feita de maneira simples, e mais detalhes sobre o formato do arquivo OBJ pode ser visto em [3].

Objeto

Dentro de um arquivo OBJ exportado pelo Blender, podem existir vários sub-objetos, como por exemplo o corpo da nave, e a cabine. Cada objeto destes contém uma lista de vértices, normais e as faces, que dizem quais os vértices que devem ser ligados em ordem. Possui a referência do material associado.

Orbita

Guarda informações sobre a órbita de um planeta, como raio, centro de translação, velocidade e a rotação acumulada até então.
Tem o papel de calcular o próximo ângulo de rotação, através d função *atualiza*, e de desenhar o trajeto da órbita (em forma de elipse) usando cubos pequenos.

Planeta

Representa um planeta. Guarda informações sobre o planeta, como raio, textura, velocidade de rotação sobre si mesmo, órbita, e em especial à terra, informações da órbita e textura da lua, já que necessita das posições x e z da mesma.

A função desenhaLua é chamada pela Terra, passando sua posição calculada, visto que é necessário saber isto para fazer a órbita correta da Lua. É semelhante à função *desenha*, em que seta a textura da esfera a ser desenhada, chama as funções de rotação, translação na ordem correta, e desenha a esfera, utilizando as funções do GLUT para quádricas, habilitando texturas.

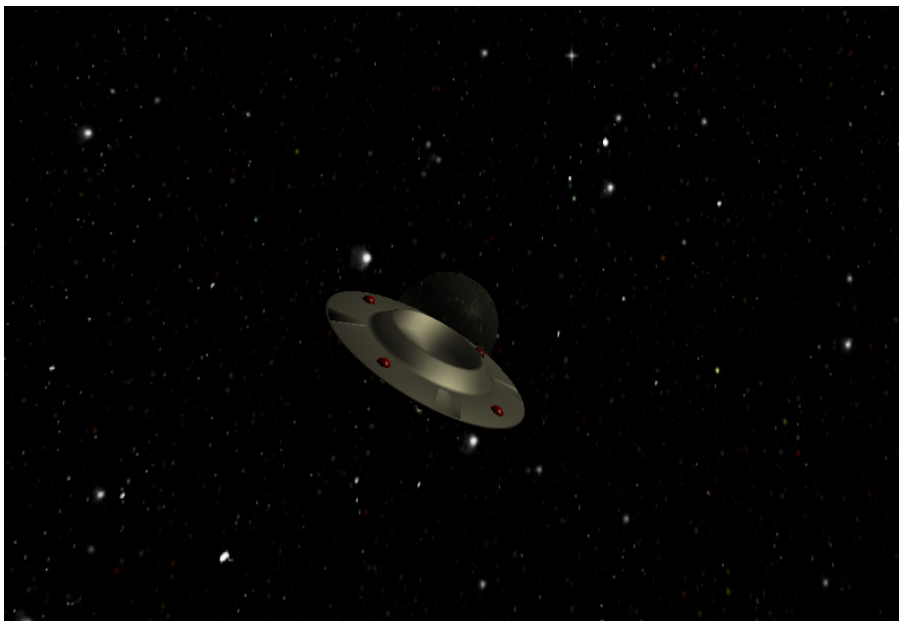
A função *desenha* faz mais uma coisa: se o planeta tiver órbita, então manda desenhá-la.

Para o cálculo da próxima posição na órbita, é realizado o mesmo cálculo já apresentado acima, na classe Anel.

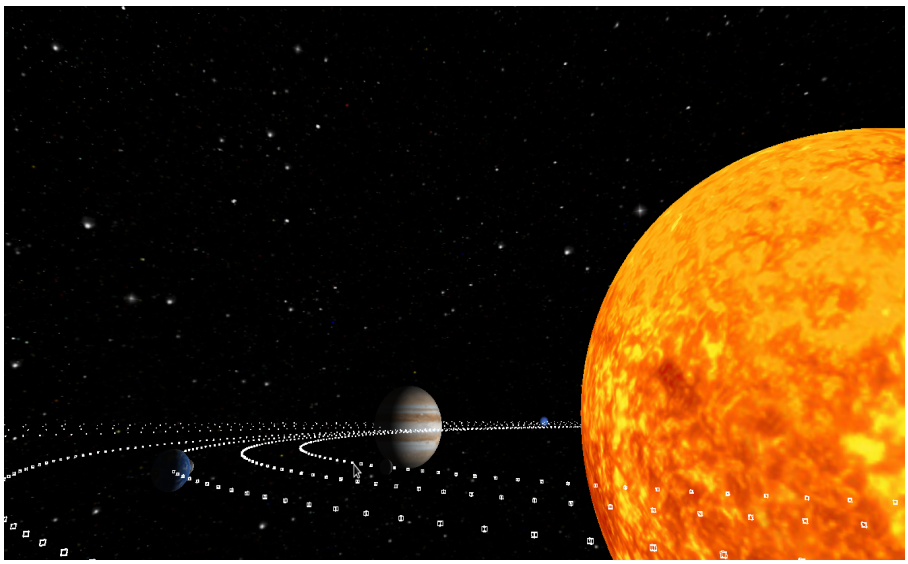
Textura

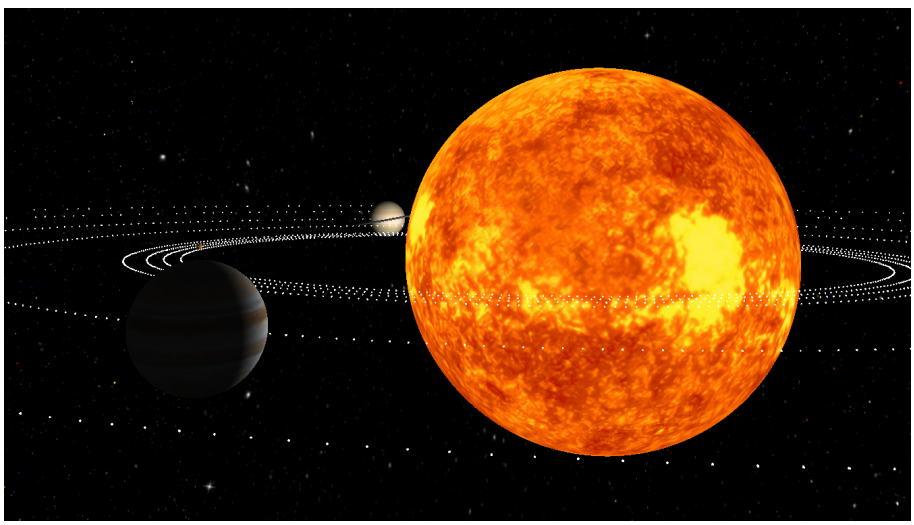
Classe para abstrair o carregamento de uma textura. Utilizamos a biblioteca STB Image para o carregamento de images, a fim de podermos utilizar arquivos JPG e PNG.
Aqui habilitamos configurações padrões de textura e mipmaps.

Imagens do resultado final

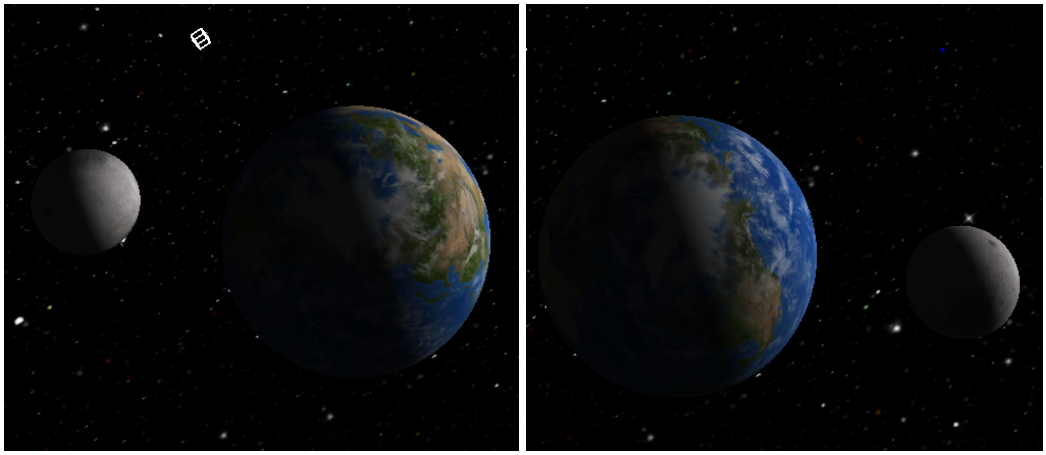


A nave possui a cabine como um globo de vidro, como pode ser visto na imagem.





O percurso percorrido pelos planetas é elíptico, como pode ser visto.



A terra gira em torno do Sol, e a lua acompanha-a girando em torno da terra, mostrando sempre a mesma face.

Referências:

[1] http://www.enchantedlearning.com/subjects/astronomy/planets/
[2] https://github.com/nothings/stb
[3] https://en.wikipedia.org/wiki/Wavefront_.obj_file
[4] http://stackoverflow.com/questions/16593758/earth-tilt-opengl-c/16594168#16594168