

**Multimídia**

**E**

**Realidade Virtual**

**Túlio Borges Contiliani 201504130456**

**Luan Carlo de Souza 201608069397**

**Marcos Mota 201702533549**

**Bruno Apolinario 201607295938**

**Bruno Albuquerque 201608091228**

# Introdução

A ideia inicial que nosso grupo teve foi de fazer um sistema solar, e que nos planetas faríamos as especificações do trabalho, como objetos animados em torno do planeta e que estes também emitissem luz conforme circulam o planeta para podermos observar seu comportamento. O resultado final de nosso projeto foi um sistema solar que possui proporções similares as reais, mas com escala reduzida, possuindo também todos os requisitos do trabalho.

# Estrutura Geral do Programa

O programa é arquiteturizado em um arquivo principal e três classes:

* **Main.cpp**
  + **Método teclado:** Contém rotinas para controle de input.
  + **Método display:** Contém todos os objetos/animação/variáveis.
  + **Método resize:** Mantém o view atualizado na alteração da janela.
  + **Método MenuSpeed:** Contém rotinas do menu de velocidade.
  + **Método MenuPrincipal:** Necessário para instanciar menu.
  + **Método CriaMenu:** Cria o menu do botão direito.
  + **Método main:** Contém todos as rotinas de inicialização bem como as rotinas de chamadas de método.
* **Classe Planeta.h**
  + **Método drawPlanet:** a partir do valor das variáveis, gera um planeta.
  + **Método setMatBright:** Seta a textura do planeta como brilhosa.
  + **Método setMatFrosted:** Seta a textura do planeta como fosca.
  + **Método setMatSun:** Especialmente seta a textura do sol (brilhosa).
  + **Método drawRing:** Desenha um anel no planeta.
  + **Método drawSquareSpaceship:** Desenha uma espaço-nave quadrada em órbita do planeta.
  + **Método drawTorusSpaceship:** Desenha uma espaço-nave triangular em órbita do planeta.
  + **Método drawMoon:** Desenha uma lua.
* **Classe Iluminacao.h**
  + **Método setColorRed:** Seta a cor da luz como vermelha.
  + **Método setColorGreen:** Seta a cor da luz como verde.
  + **Método setColorBlue:** Seta a cor de luz como azul.
  + **Método createLight:** Recebe como parâmetro um integer que, a partir disso, cria uma fonte de luz. Caso o parâmetro passado seja 0, será criado uma luz com as propriedades do sol. Caso não, será criado uma direcionalLight.
* **Classe camera.h**
  + **Método createCamera:** Cria a câmera com todas as funções.

Por convenção, todas as classes têm em seu corpo os getters e sets, originalizado da orientação a objeto.

# Detalhes de Destaque

Um dos detalhes que precisamos citar é a sequência de comandos na criação dos planetas. É importante manter a seguinte sequência para que tudo fique em sincronia total (Rotação, posição, iluminação e etc.) Desrespeitando essa sequência, não conseguimos obter o resultado desejado de um planeta.

A sequência ideal encontrada para a sincronia é:

1. Instancie o objeto.
2. Sete sua textura.
3. Sete seu tamanho.
4. Sete a velocidade de rotação em torno do próprio eixo.
5. Sete a velocidade de rotação em torno do sol.
6. Sete a posição em relação ao universo.
7. Sete a cor.
8. Desenhe o planeta.
9. Desenhe um objeto em sua orbita (**Importante:** para manter a sincronia, é importante cada planeta ter no máximo um objeto em orbita).

**Exemplo prático:**

Planeta Terra;

Terra.setMatBright();

Terra.setSize(var);

Terra.setRotation(var);

Terra.setTranslation(var);

Terra.setPos(var);

Terra.setColor(var);

Terra.drawPlanet();

Terra.drawSquereSpaceship();

# Manual do Usuário

Para poder controlar a aplicação configuramos algumas teclas do teclado para movimentar a câmera. Utilizamos as teclas “w”, “a”, “s”, “d”, “o” e “l”.

* “**W**” - é fazer com que a câmera se movimente para a posição no eixo “y” possibilitando que vejamos o sistema solar de cima.
* ”**A**” - faz com que a câmera se movimente para a esquerda.
* “**S**” - faz com que a câmera vá para baixo do sistema solar possibilitando que vejamos ele pelo ângulo inferior.
* ”**D**” - move a câmera para o lado direito.
* “**O**” – zoom in.
* “**L**” - zoom out.

# Instruções de Compilação e Execução

No projeto, utilizamos OPENGL e GLUT. São necessários algumas configurações para poder compilar. Ao contrário disso, para executar é necessário somente um sistema operacional Windows para executar o .exe do simulador, uma vez que ele é feito em C++.

* **Para compilar**
  + Necessita da instalação de um compilador de C++. No caso, nós usamos o GCC.
  + Instalar a biblioteca GLUT pois utilizamos a biblioteca GL/glut.h