Relatório de Computação Gráfica

Trabalho 2

Gabriel Henrique Campos Scalici 9292970

Keith T. Sasaki 9293414

Enunciado do trabalho: Confeccionar um jogo com a openGL que simule o jogo Space Invader.

O jogo deve conter uma nave móvel (percorre a tela no sentido horizontal de acordo com a tecla pressionada pelo usuário) que atira lasers (mísseis) em inimigos que ficam percorrendo a tela descendo em direção à nave.

Plataforma utilizada: O referido trabalho foi confeccionado no sistema operacional MacOS e Ubuntu(Linux). As IDEs utilizadas foram, o ambiente Xcode (no macOS) e Sublime (no Linux), o qual fazem uso da linguagem C++ para a implementação do trabalho. Para compilar foi usado o botão play automático da IDE e também o uso do arquivo Makefile, para caso o usuário queira rodar via linha de comando.

Tecnologias externas: Não foram usadas outras tecnologias além das funções presentes no OpenGL.

Principais dificuldades: A principal dificuldade que tivemos durante a criação do jogo foi a parte que envolvem as colisões do laser que sai da nave com os inimigos, pois da forma como implementamos o movimento dos inimigos(auxílio de variáveis em cada matriz para desenhar polígonos), foi difícil o armazenamento da posição de cada um.

Outra parte que também tivemos bastante problema foi a quantidade de lasers que a nave principal pode atirar, pelo fato de que iniciar o laser e o transladar quando o usuário atira, deve ser feito tudo somente em uma primitiva geométrica. Assim fizemos uma lógica para que possamos usar a mesma bala várias vezes.

Divisão de tarefas: O trabalho inteiro foi construído praticamente junto, onde um realizava as pesquisas sobre as funções corretas (e explicações) e como fixar erros que impediam a compilação, principalmente porque

trabalhamos em dois sistemas operacionais diferentes. Trocando de função diversas vezes durante o processo, para que pequenos erros e detalhes pudessem ser corrigidos, de acordo com o tempo disponível de cada um.

Resumidamente a área de código escrito:

Keith : Movimentação dos inimigos, sistema de colisões do míssel com o inimigo (Fazendo com que o inimigo fique fora de jogo), mísseis que o usuário irá atirar..

Gabriel: Estrutura dos inimigos, estrutura da nave do jogador, estrutura do míssel, parte que envolve ViewPort do projeto, correção das velocidades.

Tempo: Para a realização deste trabalho foi preciso trabalhar por 6 dias, sendo que houve dias que não houveram tanto tempo de dedicação, porém demoramos para construir algo relativamente simples pois usamos como oportunidade de estudo sobre OpenGl, interrompendo diversas vezes o trabalho para discussões.

Trecho de código que o grupo julga mais importante: Dentre os tópicos abordados no projeto, achamos que o mais importante do trabalho foi a movimentação, morte e análise de colisão do tiro com os inimigos. Um trecho da colisão do inimigo mais próximo da área esquerda da tela:

```
if(vivos[0])
    glColor3f(1.0f, 1.0f, 0.88f);
  else
    glColor3f(0.18f, 0.31f, 0.31f);
  glBegin(GL_POLYGON);
    glVertex2f(moveInimigox,fileira1y + alturaInimigo);
    glVertex2f(moveInimigox, fileira1y);
    glVertex2f(moveInimigox + larguraInimigo, fileira1y);
    glVertex2f(moveInimigox + larguraInimigo, fileira1y + alturaInimigo);
  glEnd();
  inimigos[0].x = moveInimigox;
  inimigos[0].y = fileira1y;
```

A parte que envolve a lógica para movimentação dos inimigos:

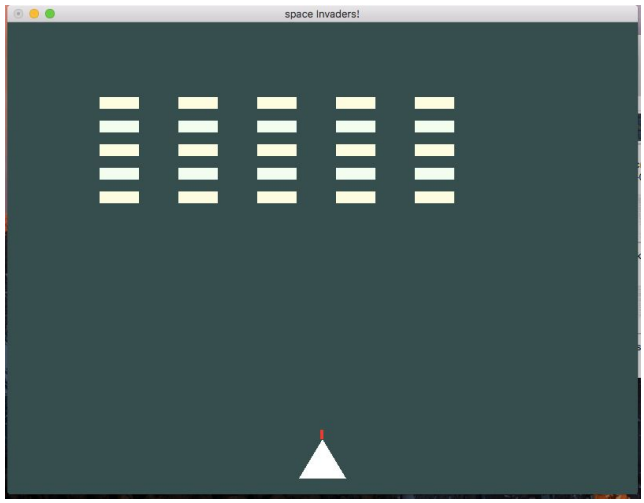
```
void moveInimigos(int passo){
  if(direcaoX == DIREITA){
    moveInimigox += passo;
    if(moveInimigox + distanciaX*4+larguraInimigo >= 799){
      fileira1y -= 30;
      fileira2y -= 30;
      fileira3y -= 30;
      fileira4y -= 30;
      fileira5y -= 30;
      direcaoX = ESQUERDA;
    }
  }
  else{
    moveInimigox -= passo;
    if(moveInimigox == 0){
      direcaoX = DIREITA;
      fileira1y -= 30;
      fileira2y -= 30;
      fileira3y -= 30;
      fileira4y -= 30;
      fileira5y -= 30;
    }
  }
```

E também a análise das variáveis para saber se o inimigo chegou ao fim da tela, para que o game over seja anunciado:

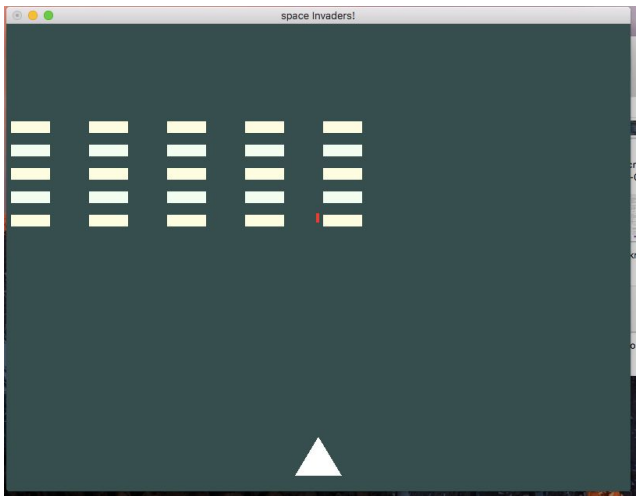
```
if(chegouNoFinal()){
  glClearColor(1.0, 0.0, 0.0, 0.0);
      glClear(GL_COLOR_BUFFER_BIT);
  passo = 0;
  glutSpecialFunc(NULL);
}
```

Demonstração de funcionalidades:

Temos a tela inicial do game:



O tiro quando é disparado:



E o inimigo quando é acertado e eliminado do jogo: