Trabalho 1 - Computação Gráfica
Jogo Asteroides

Bernardo Luiz Padilha Zamin
Leonardo Oliveira
Professor Márcio Pinho

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Caixa Postal 1429 – 90.619-900 – Porto Alegre – RS – Brazil

Abstract. *This report aims to address the requirements specified by the professor in the first assignment of the Computer Graphics. It provides an objective and clear overview of the requirements and how we met them.*

Resumo. *Este relatório busca contemplar o atendimento dos requisitos especificados pelo professor no trabalho 1 da disciplina de Computação Gráfica. Ele disponibiliza uma visão objetiva e clara dos requisitos e como fizemos para atende-los.*

1. Introdução

Neste trabalho, vamos explicar como atendemos às solicitações do professor Márcio Pinho[1] na disciplina de Computação Gráfica. Vamos abordar desde entender e colocar em prática soluções adequadas. Vamos falar sobre modelagem de objetos e como tudo isso se junta para criar imagens legais. Também vamos discutir as ferramentas que usamos e como lidamos com os desafios ao longo do caminho. No final, queremos mostrar não só que conseguimos atender às exigências do professor, mas também aprender e melhorar nossas habilidades nessa área empolgante da computação.

2. Criação de Instâncias de personagens e figuras

Após carregarmos os modelos dos personagens utilizando a função *CarregaModelos()*, utilizamos a função *CriaInstancias()*, disponibilizada pelo professor, para instanciarmos o disparador (personagem principal), as naves inimigas, as figuras representativas de vida e os tiros do disparador e de seus inimigos. As imagens do disparador (figura 4), assim como as dos inimigos (figura 5) e das vidas (figura 6) são de nossa própria autoria e as fizemos utilizando arquivos *.txt* e ferramentas como o *Excel* para modelarmos os arquivos. Também criamos um cenário (figura 3) para o jogo utilizando as mesmas ferramentas. As telas de inicio do jogo e de *Game Over* são representadas pelas figuras 1 e 2, respectivamente.

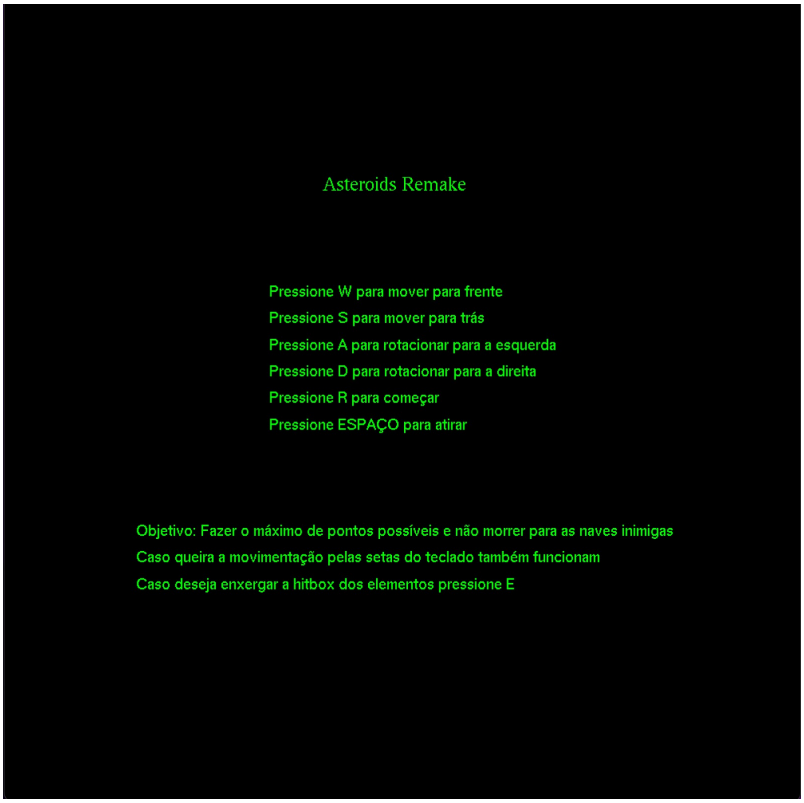

Figura 1. Tela de inicio

Figura 2. Tela de ”Game Over”

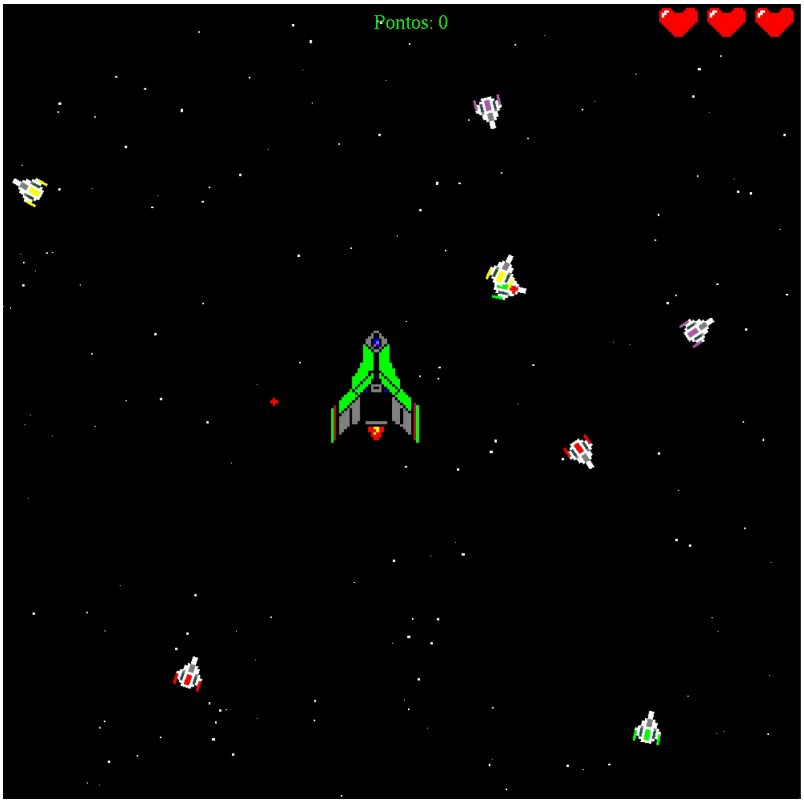

Figura 3. Cenário do Jogo

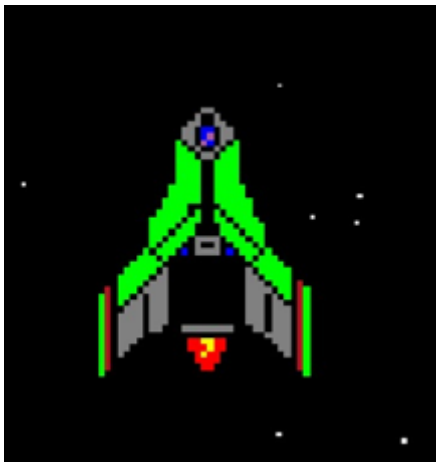

Figura 4. Disparador

Figura 5. Nave Inimiga

Figura 6. Vidas

3. Movimentação e disparos do disparador e naves inimigas

Para o desenvolvimento das movimentações do jogador principal (disparador), utilizamos a função *keyboard()*, utilizando as teclas **W, A, S e D** e **Espaço** para o disparo do tiro. Já para as naves inimigas desenvolvemos movimentações e disparos aleatórios adaptados para um jogo limpo. O disparador tem um limite de 10 tiros no máximo a cada certo curto período de tempo, no qual adequamos para tornar melhor a jogabilidade. Aqui estão as funções das teclas envolvidas:

- Tecla 'W': Movimentação do disparador dianteira
- Tecla 'S': Movimentação do disparador traseira
- Tecla 'D': Rotação do disparador para a direta
- Tecla 'A': Rotação do disparador para a esquerda
- Tecla espaço : Disparo(s) do disparador
- Tecla 'E': Habilita/desabilita a impressão dos envelopes dos personagens
- Tecla 'Q' e Escape: Para a execução do programa

3.1. Colisões e Mortes

O disparador morre somente se for atingido três vezes por naves inimigas. O disparador pode eleminar os inimigos além de atingindo-os com os tiros, mas tambem colidindo com eles, onde esta função é implementada em *TestaColisaoNaves()*. Essa colisão dos tiros, é calculada pelas funções *TestaColisaoTirosInimigos(), TestaColisaoTirosJogador() e ColisaoTiroPersonagem()*. Também quando o disparador é atingido, uma vida é perdida, removendo um coração da tela. Todo calculo de colisão é chamado na função *AtualizaJogo()*, que é chamada pela função *AtualizaPersonagens()* que por fim, é chamada pela função *display()* assim sempre atualizando a tela.

3.1.1. Explicando a função TestaColisaoTirosInimigos()

Se o tiro disparado for do disparador e há colisão deste tiro com o inimigo, então o inimigo morre, sumindo do jogo e gerando um novo inimigo aleatorio no mapa. Cada inimigo morto acumula um total de 20 pontos ao jogador.

3.1.2. Explicando a função TestaColisaoTirosJogador()

Se o tiro de uma nave inimiga e for disparador e houver colisão com o jogador principal, então é tirada uma vida do jogador e é feita uma verificação se o número de vidas é igual a zero. Caso essa verificação for positiva o jogo acaba e a tela de *gameover* aparece.

3.1.3. Explicando a função ColisaoTiroPersonagem()

A função ColisaoTiroPersonagem é utilizada na função *TestaColisaoTirosInimigos()* para testar a detecção de colisões usando *bounding boxes* (envelope). Essa função verifica se um tiro disparado colide com um inimigo. Para isso, calcula as coordenadas mínimas e máximas dos envelopes tanto do tiro quanto do inimigo e verifica a sobreposição desses envelopes. Se as *bouding boxes* se sobrepõem em ambas as direções x e y, então ocorre uma colisão; caso contrário, não há colisão

3.1.4. Explicando a função TestaColisaoNaves()

A função *TestaColisaoNave* verifica se a nave do jogador colide com alguma nave inimiga. Se houver colisão, o jogador ganha os pontos, a nave inimiga desaparece sendo eliminada e a função *GeraNovosInimigos* é chamada para gerar novos inimigos aleatoriamente.

4. Disparo de tiros

O disparo de tiros feitos pelo disparador são feitos pela frente da nave, linearmente. Utilizamos uma lógica de recarregamento do disparador para que ele não dispare mais de 10 tiros seguidos, sem esperar com que o outro acabe, com um tempo de recarga de 1 segundo e meio. Para ajustar o tiro para que saia pela frente da nave, utilizamos o cálculo do ponto/posição da ponteira do disparador junto com um *offset*(ajustado manualmente, de acordo com as posições da nave), modelando assim a posição do tiro com base na posição e na rotação em que o disparador se encontra. Eles são implementados pela função *disparatirojogador()*, onde o cálculo de cada ponteira da nave é realizado na própria função. Os disparos inimigos são aleatórios e a velocidade do tiro é menor que o tiro do disparador (jogador principal), para que o jogo tenha uma boa jogabilidade.

5. Lógica e características do jogo

O objetivo do disparador é eliminar o máximo de inimigos possível, acumulando o maior número de pontos. O jogo não possui um final definido, pois naves inimigas surgem constantemente, criando uma dinâmica de competição por recordes entre os jogadores. À medida que o jogo avança, a quantidade e a velocidade dos inimigos aumentam, desafiando continuamente o jogador. O jogador começa com um número limitado de vidas, e ao ser atingido por tiros inimigos, perde uma vida. O jogo termina quando todas as vidas são perdidas. Controles simples e responsivos permitem ao jogador focar na ação, enquanto uma interface limpa exibe pontuação, vidas restantes.

6. Link para o vídeo

Visualizar Vídeo
ou
https://www.youtube.com/watch?v=3JMjRWGcleo

Referências

[1] “Márcio pinho, professor da disciplina de computação gráfica na pucrs,” 2024.