TÉCNICAS DE PROGRAMAÇÃO II

TRABALHO 2

Descrição:

O objetivo do trabalho 2 é desenvolver um jogo estilo *shoot 'em up* em Java, onde o jogador possa controlar uma espaçonave e destruir as espaçonaves inimigas que surgirem.

A Figura 1 ilustra a interface e os elementos básicos do jogo.

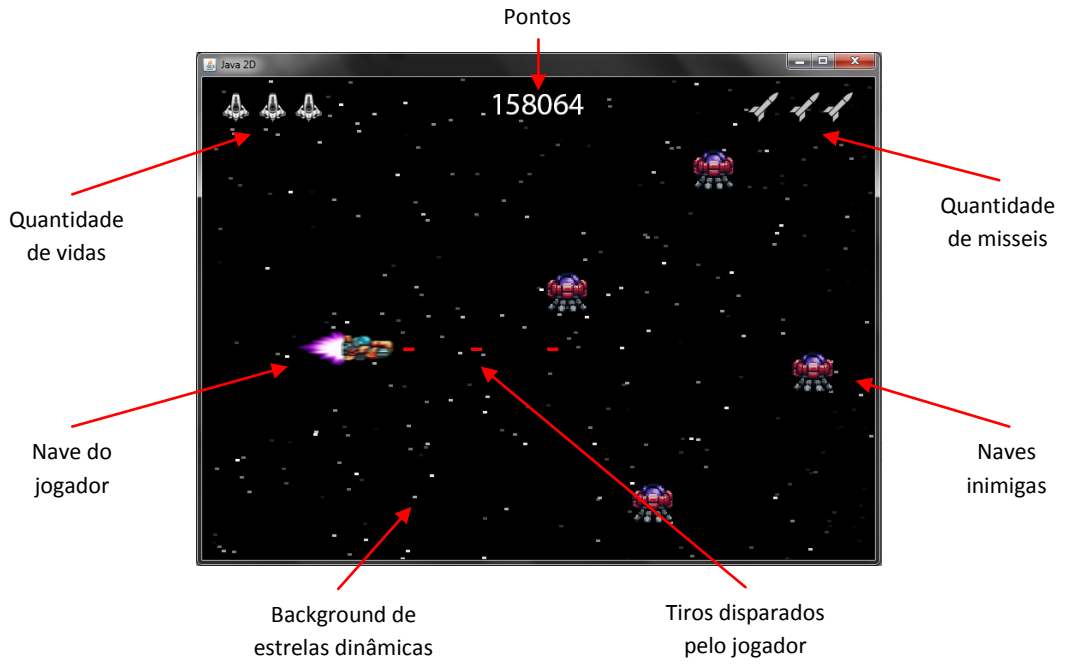


Figura 1: Interface e Gameplay.

O jogo deve ser desenvolvido seguindo o paradigma de programação orientado a objetos, onde os diversos elementos da aplicação devem ser organizados e implementados em classes.

Requisitos Básicos:

- O jogador deverá poder controlar a sua nave pelo teclado em todas as direções (right, left, up e down).

- O jogo não deve permitir que a nave controlada pelo jogador saia da região visível da tela.

- As estrelas do background devem ser geradas aleatoriamente e movimentar-se dinamicamente de acordo com a velocidade da nave e distancia da estrela (criando a sensação de profundidade). Estrelas distantes devem mover-se mais lentamente e serem menos brilhantes, enquanto que estrelas próximas devem mover-se mais rapidamente e serem mais brilhantes.

- A nave controlada pelo jogador deve possuir um modo turbo que permita que a nave possa movimentar-se mais rapidamente.

- A nave controlada pelo jogador deve possuir pelo menos dois tipos de armas:

    - Tiro simples: segue uma trajetória em linha reta a partir da posição da nave até atingir um alvo ou sair dos limites da tela;

    - Míssil: segue uma trajetória na direção da nave inimiga que estiver mais próxima do jogador até atingi-la.

- O Tiro Simples é infinito, porém o Míssil é limitado. Inicialmente o jogador começa com 3 misseis. Os misseis utilizados podem ser recarregados ao pegar um item do tipo míssil que pode surgir ao destruir uma nave inimiga.

- Devem existir no jogo pelo menos 4 tipos de inimigos:

Nome	Comportamento	Imagem
RedUfo	Movimenta-se apenas horizontalmente para frente e não possuí nenhuma arma. Um tiro é suficiente para ele ser destruído.	
GreenFire	Movimenta-se apenas horizontalmente para frente e possui uma arma que dispara tiros em determinados intervalos de tempo. Um tiro é suficiente para ele ser destruído.	
BigBang	Movimenta-se diagonalmente (cima e frente ou baixo e frente) e não possuí nenhuma arma. São necessários 3 tiros para destruí-lo ou 1 míssil.	
DeathFish	Movimenta-se de forma ondular para frente e possui uma arma que dispara tiros em determinados intervalos de tempo. São necessários 5 tiros para destruí-lo ou 1 míssil.	

- O jogo deve ser capaz de gerar os inimigos aleatoriamente de acordo com um nível de dificuldade incremental. Inicialmente, apenas RedUfos devem entrar aleatoriamente na tela. Após um tempo ou após o jogador acumular um determinado número de pontos, GreenFires devem começar a aparecer. Você

deve definir qual será o critério para aumentar dificuldade do jogo e fazer com que os diferentes tipos de inimigos apareçam. Lembre-se que a quantidade de inimigos que vão aparecer também pode ser usada para aumentar a dificuldade do jogo.

- O jogo deve detectar a colisão entre a nave controlada pelo jogador e os inimigos. Em caso de colisão, ambas as naves devem ser destruídas e o jogador perde uma vida.

- O jogo deve detectar a colisão entre os tiros disparados pelo jogador e os inimigos. Em caso de colisão, a nave inimiga deve ser destruída e o jogador ganha uma determinada quantidade de pontos:

    - RedUfo: +10
    - GreenFire: +50
    - BigBang: +100
    - DeathFish: +200

- O jogo deve detectar a colisão entre os tiros disparados pelos inimigos e o jogador. Em caso de colisão, a nave do jogador deve ser destruída e o jogador perde uma vida. Se o jogador ficar sem vidas, a tela de gameover deve ser exibida.

- Após destruir um inimigo, deve existir 10% de chance de um item do tipo Míssil surgir no local onde o inimigo estava. O jogo deve permitir que o jogador possa pegar esse item (ao colidir com ele). Ao pegar o item, a quantidade de misseis disponíveis deve ser incrementada. O jogador pode ter no máximo 3 misseis ao mesmo tempo.

- O jogo deve gerenciar e exibir na tela a quantidade de vidas do jogador, a pontuação e a quantidade de misseis.

- O jogo deve ser capaz de registrar e exibir as 10 melhores pontuações conseguidas pelos jogadores. Essa informação deve ser gravada e lida de um arquivo, de modo que os recordes continuem gravados mesmo quando o programa for fechado.

- Na tela de gameover, o jogo deve exibir a pontuação conseguida pelo jogador e, se ela estiver entre as 10 melhores pontuações, o jogo deve permitir que o jogador escreva o seu nome e registre a pontuação.

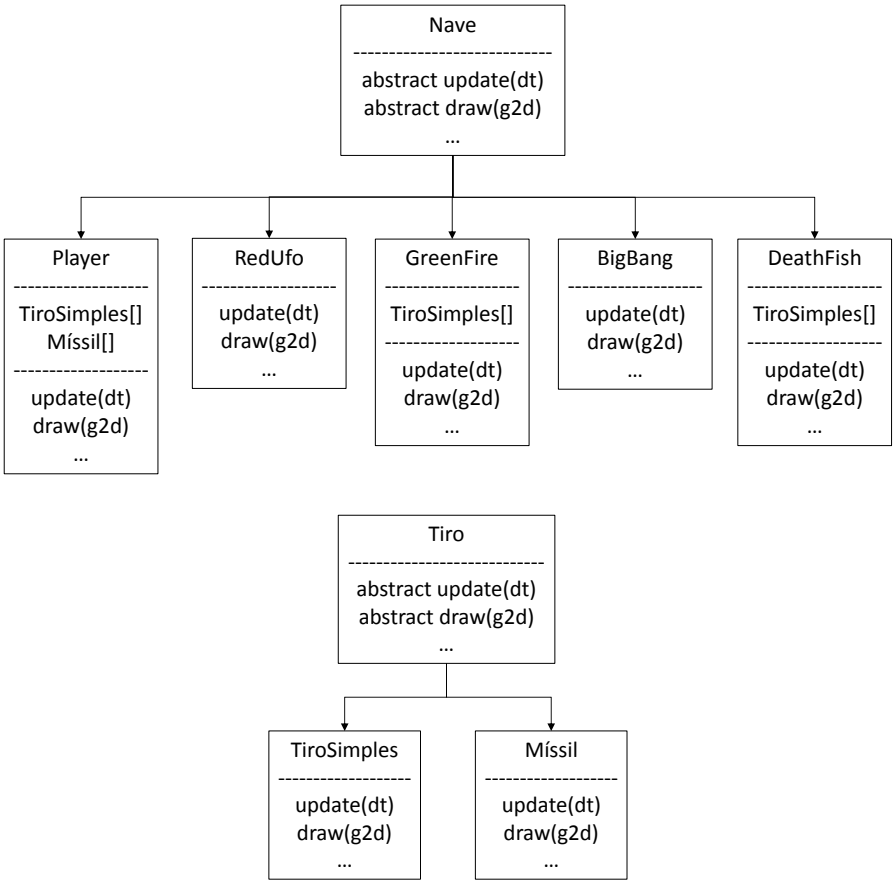
- O jogo deve possuir uma tela de título com as seguintes opções:
    - Novo Jogo: inicia um novo jogo;
    - Recordes: mostrar uma nova tela com as 10 melhores pontuações conseguidas pelos jogadores (juntamente com os nomes dos jogadores);
    - Sair: fecha o jogo.
- Você é responsável pela tarefa de tornar o jogo divertido, então está livre para modificar e melhorar todos os requisitos básicos, podendo:
    - Criar novos inimigos ou melhorar o comportamento dos inimigos definidos no enunciado;
    - Criar novas armas e power-ups para o jogador;
    - Definir uma lógica própria para incrementar a dificuldade do jogo gradativamente ou definir fases para o jogo;
    - Adicionar chefões para cada nível de dificuldade ou fase;
    - Limitar o uso do turbo de acordo com alguma lógica (tempo pré-determinado, pegar itens que permitam usar o turbo, etc.);
    - Criar outros modos de jogo, como um modo multiplayer cooperativo ou competitivo;
    - Ou qualquer outra ideia que você tiver para tornar o jogo mais divertido!
- As imagens básicas necessárias para implementar o jogo estão disponíveis no seguinte link: http://www.inf.puc-rio.br/~elima/jogos/imagens_spaceshooter.zip

Organização das Classes:

Obrigatoriamente o jogo deve ser desenvolvido seguindo o paradigma de programação orientado a objetos, onde os diversos elementos da aplicação devem ser organizados e implementados em classes.

Você é responsável por definir a estrutura e organização das classes necessárias para o jogo. Porém, pede-se que o jogo tenha no mínimo a seguinte hierarquia de classes para definir os elementos básicos do jogo:

- Nave: classe base abstrata para representar naves espaciais;
- Player: classe que herda as características da classe Nave e implementa as funcionalidades necessárias para representar e controlar a nave do jogador;
- RedUfo: classe que herda as características da classe Nave e implementa os métodos necessários para representar e definir o comportamento do inimigo RedUfo;
- GreenFire: classe que herda as características da classe Nave e implementa os métodos necessários para representar e definir o comportamento do inimigo GreenFire;
- BigBang: classe que herda as características da classe Nave e implementa os métodos necessários para representar e definir o comportamento do inimigo BigBang;
- DeathFish: classe que herda as características da classe Nave e implementa os métodos necessários para representar e definir o comportamento do inimigo DeathFish;
- Tiro: classe base abstrata para representar tiros;
- TiroSimples: classe que herda as características da classe Tiro e implementa os métodos necessários para representar e definir o comportamento de um Tiro Simples;
- Míssil: classe que herda as características da classe Tiro e implementa os métodos necessários para representar e definir o comportamento de um Míssil;



Informações Adicionais:

- O trabalho deve ser feito **individualmente**.
- Obrigatoriamente o jogo desenvolvido deverá ser apresentado durante a aula.
    - O aluno que **não comparecer** no dia da apresentação receberá nota **zero**;
    - O aluno que **não souber explicar** algo importante relacionado ao trabalho receberá nota **zero**.

Forma de Avaliação:

Será avaliado se:

(1) O trabalho atendeu a todos os requisitos especificados anteriormente;
(2) O código foi devidamente implementado, organizado e documentado;
(3) A interface com o usuário atende aos requisitos básicos de usabilidade;
(4) O trabalho foi apresentado corretamente em sala de aula;

Data de Entrega e Apresentação:

25/11

Forma de Entrega:

O programa deve ser apresentado na aula do dia 25/11 (terça) e enviando até o mesmo dia pelo Moodle.

Trabalhos entregues atrasados perderam 0.5 pontos para cada dia de atraso.