Universidade do Porto
Faculdade de Engenharia
FEUP

Raiden

Relatório Final

Laboratório de Programação Orientada a Objetos
2º ano do Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

João Monteiro – 201104369 – ei11055@fe.up.pt
Miguel Tavares – 201105547 – ei11069@fe.up.pt

9 de Junho de 2014

adfa, p. 1, 2011.

1 Introdução

Este relatório tem por objetivo documentar o trabalho desenvolvido no projeto em questão, que consistiu no desenvolvimento do jogo de arcade Raiden, na linguagem de programação Java, seguindo os paradigmas da Programação Orientada a Objetos.

O projeto teve por objetivo demonstrar a aplicação dos conceitos lecionados na unidade curricular, com particular destaque para a modelação UML, a utilização da biblioteca Java Swing e o desenvolvimento de código seguindo padrões de desenho.

Serão de seguida apresentados um pequeno manual de utilização do programa, a conceção e implementação do programa em termos de arquitetura e bibliotecas utilizadas. Pretende-se assim dar uma ideia sucinta acerca do que trata o projeto, tal como o trabalho desenvolvido.

2 Manual de utilização

Abrindo o executável, abrir-se-á a janela correspondente ao jogo. No menu disponibilizado, será possível ao utilizador escolher entre jogar um novo jogo ou sair do programa. No caso de o jogador optar por iniciar uma nova partida, ser-lhe-á dado a escolher qual o nível de dificuldade do jogo, entre fácil, médio e difícil.

O jogador terá controlo sobre uma nave espacial, tendo como obstáculos asteroides, cujo impacto reduz a vida (HP) do jogador, e inimigos, naves controladas pelo programa que lançam projéteis tal como o jogador. A nave será controlada através das setas do teclado, sendo a barra de espaço utilizada para disparar tiros.

O jogador poderá apanhar power-ups ao longo de um nível. Esses podem ter efeitos variados: aumento do dano, aumento da velocidade, regeneração de vida, incremento da pontuação e até invulnerabilidade temporária.

Haverá três dificuldades de jogo: fácil, médio e difícil. Dependendo dessa dificuldade, haverá alterações na densidade de asteroides em cada nível, bem como na inteligência nas reações dos inimigos.

O jogo poderá ser jogado por um ou dois jogadores. No caso de jogarem duas pessoas, considera-se que jogam na mesma equipa; isto é, apesar de terem pontuações diferentes, os projéteis de um não afetam a nave do outro, e a colisão entre as naves também não tem qualquer consequência.

3 Conceção e implementação

Anteriormente à implementação propriamente dita do programa em Java, foi desenhada a estrutura do programa de maneira a perceber antecipadamente o seu funcionamento. Esse desenho inicial foi sendo alvo de refactoring, à medida que foram sendo adicionadas novas funcionalidades, a interface gráfica do programa e os métodos de teste. O resultado final do desenho é apresentado de seguida.

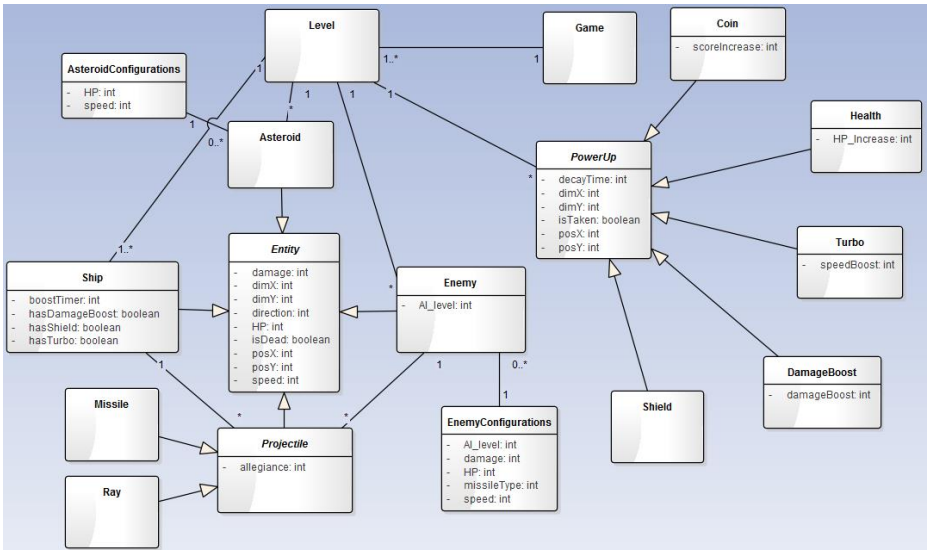

Figura 1 - Diagrama UML de Classes

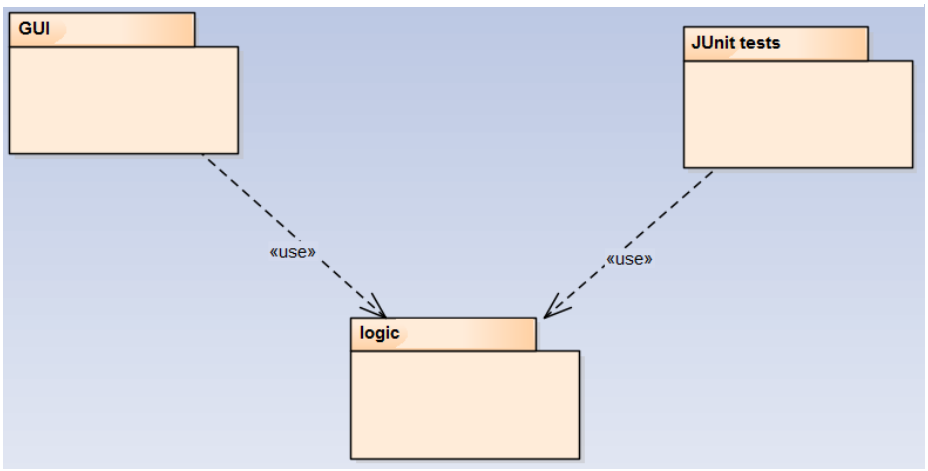

Figura 2 - Diagrama UML de packages

O código desenvolvido no projeto tem presentes vários padrões de desenho característicos. Estes padrões visam forçar uma comunicação mais eficaz entre os programadores, redução da complexidade do código e redução no tempo de aprendizagem de novas classes ou bibliotecas. Assim sendo, no

código desenvolvido, podem ser detetados vários padrões de desenho. Nomeadamente Factory Method na superclasse Entity e suas subclasses, ao ser definida uma classe abstrata para elementos com certas propriedades em comum, sendo depois definido o comportamento diferente na própria subclasse; Template Method, nas funções paint() e addNotify() que implementam métodos definidos em superclasses(ou interfaces); Strategy é utilizado ao nível das funções de movimento da nave, por disponibilizar funções que executam operações específicas sobre o objeto da sua responsabilidade (alterar a sua posição);

4 Conclusão

A implementação seguiu proximamente a conceção sugerida no diagrama UML de classes. A maior diferença a ressaltar foi a omissão da classe Level. Originalmente, um Game seria composto por um ou mais Levels, que iam sendo desbloqueados progressivamente. No entanto, por falta de tempo, foi decidido cortar essa funcionalidade e incorporaram-se na classe Game os métodos e atributos de ambas as classes, simplificando a estrutura do jogo.

Possíveis melhorias do trabalho incluiriam a introdução do registo de highscores, a introdução de som associado aos diferentes eventos do jogo, a publicação do jogo numa plataforma móvel(Android) ou a execução do jogo em rede com vários jogadores, sendo a competição feita para ver quem obtém o maior número de pontos num determinado período de tempo. Também seria de pensar a introdução de diferentes tipos de inimigos.

O trabalho desenvolvido foi dividido de forma mais ou menos equitativa pelos dois elementos do grupo.

5 Referências

5.1 Bibliografia

Apontamentos das aulas teóricas.

5.2 Links web

Tutoriais Oracle Java.