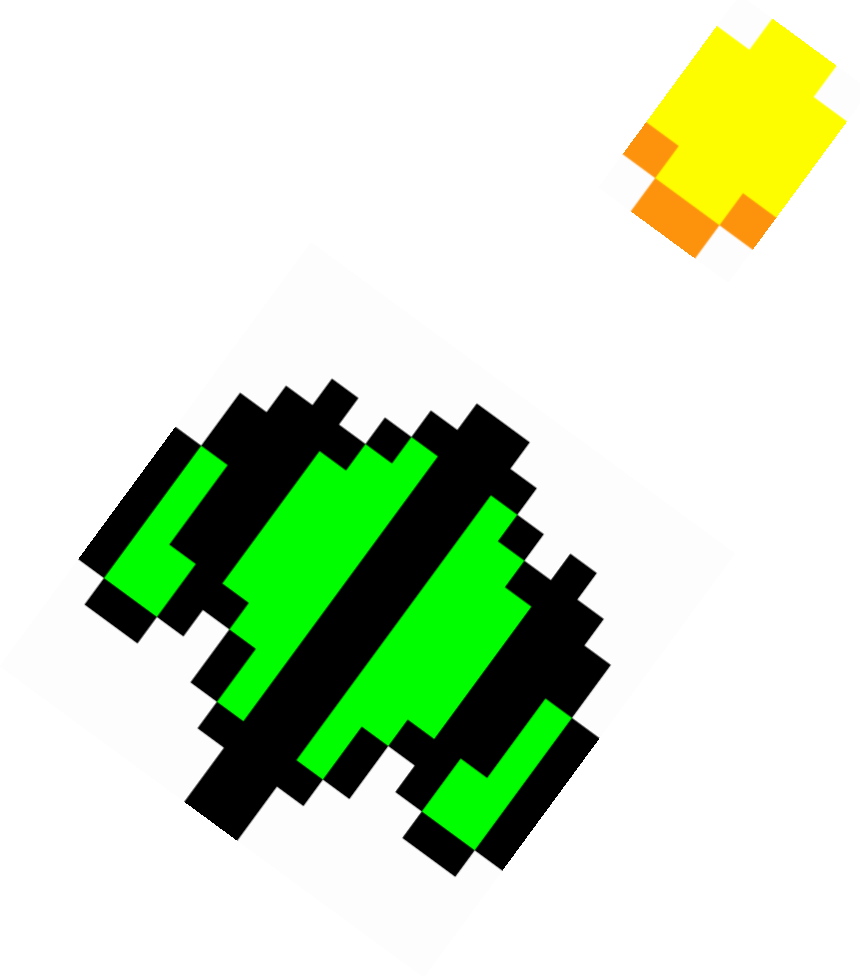
FEUP - MIEIC

LPOO

Kernel Defender

Professores: Ademar Aguiar, Nuno Flores



Diogo Miguel Sousa Barroso, ei11105
Miguel Geraldes Antunes Mendes, ei11058

Introdução

Objetivo do Relatório

Este relatório tem como objetivo a apresentação da planificação e do desenvolvimento do jogo Kernel Defender, para a Unidade Curricular de Laboratório de Programação Orientada a Objetos (LPOO), no ano curricular 2013/14. Com isto, é pretendido que o leitor comum possa compreender a estrutura de todo o programa, através de diagramas UML explicativos. Também serão dadas instruções quanto à execução do mesmo, para que não tenha de explorá-lo para usufruir do mesmo.

Objetivo do Programa

O *Kernel Defender* é um jogo que mistura alguns componentes de *Shooter* e *Tower Defense*. O objetivo é impedir os vírus de chegarem ao *Kernel* (localizado no canto superior esquerdo da janela, representado por um quarto de círculo). O jogador (anti-vírus, presente na capa), contará com as suas armas para os derrotar. Para direcionar algumas armas precisará de apontar na direção dos vírus, enquanto que outras terão projéteis guiados. Optámos por *Pixel Art* para um visual minimalista e simples, divertindo o jogador ao mesmo tempo. A *library* usada para o desenvolvimento do jogo foi a *libgdx*.

Manual de Utilização

Arranque do Programa

Para correr o jogo, bastará fazer um duplo clique no ficheiro executável. Deverá abrir uma janela ativa, não maximizada, de resolução 1280x720.

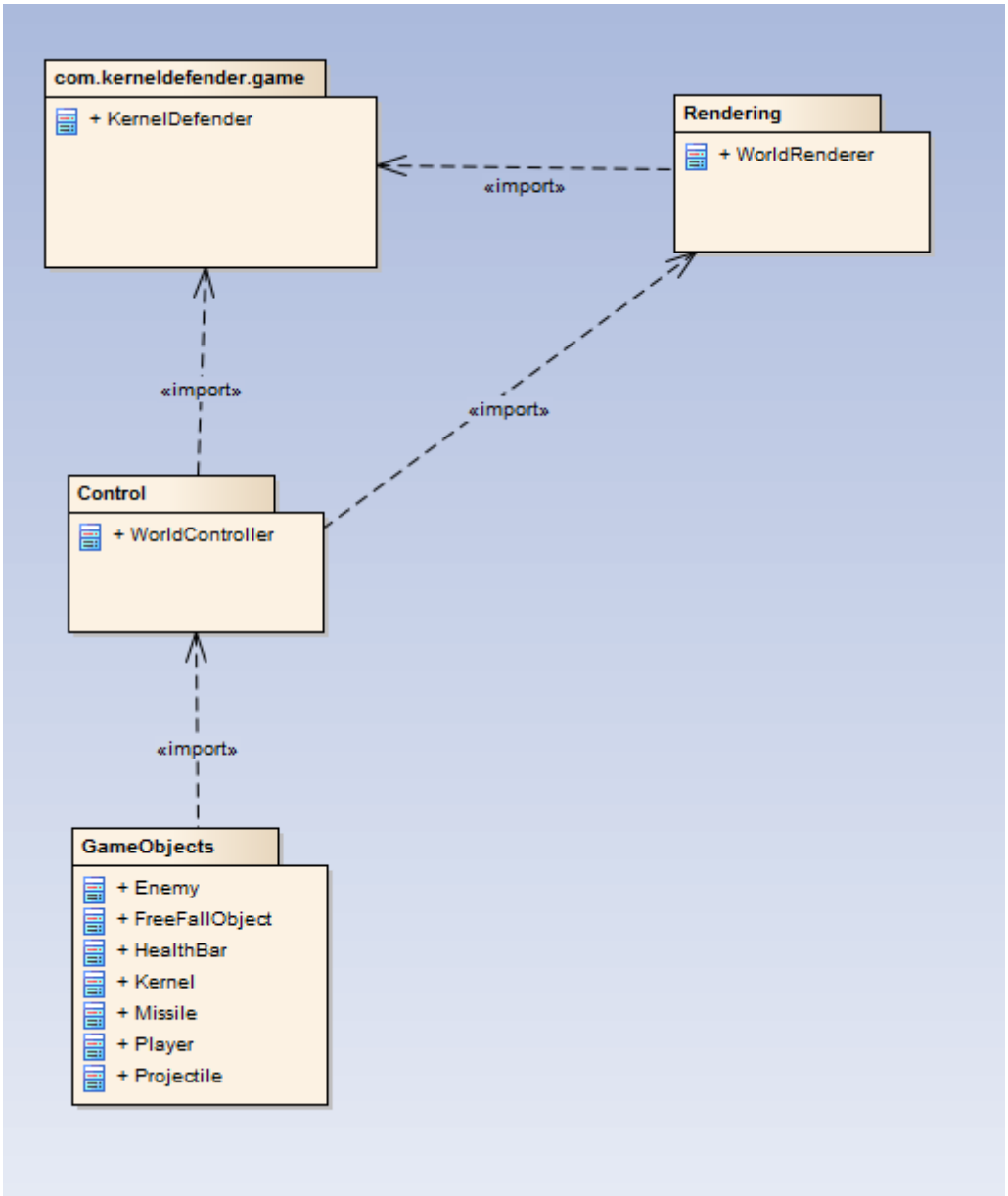
Modo de Utilização

Para jogar, os comandos são:
A + S move o jogador no sentido do relógio
W + D move o jogador no sentido inverso ao do relógio
Botão esquerdo do rato dispara arma normal
Botão direito do rato dispara mísseis auto-guiados

Conceção e Implementação

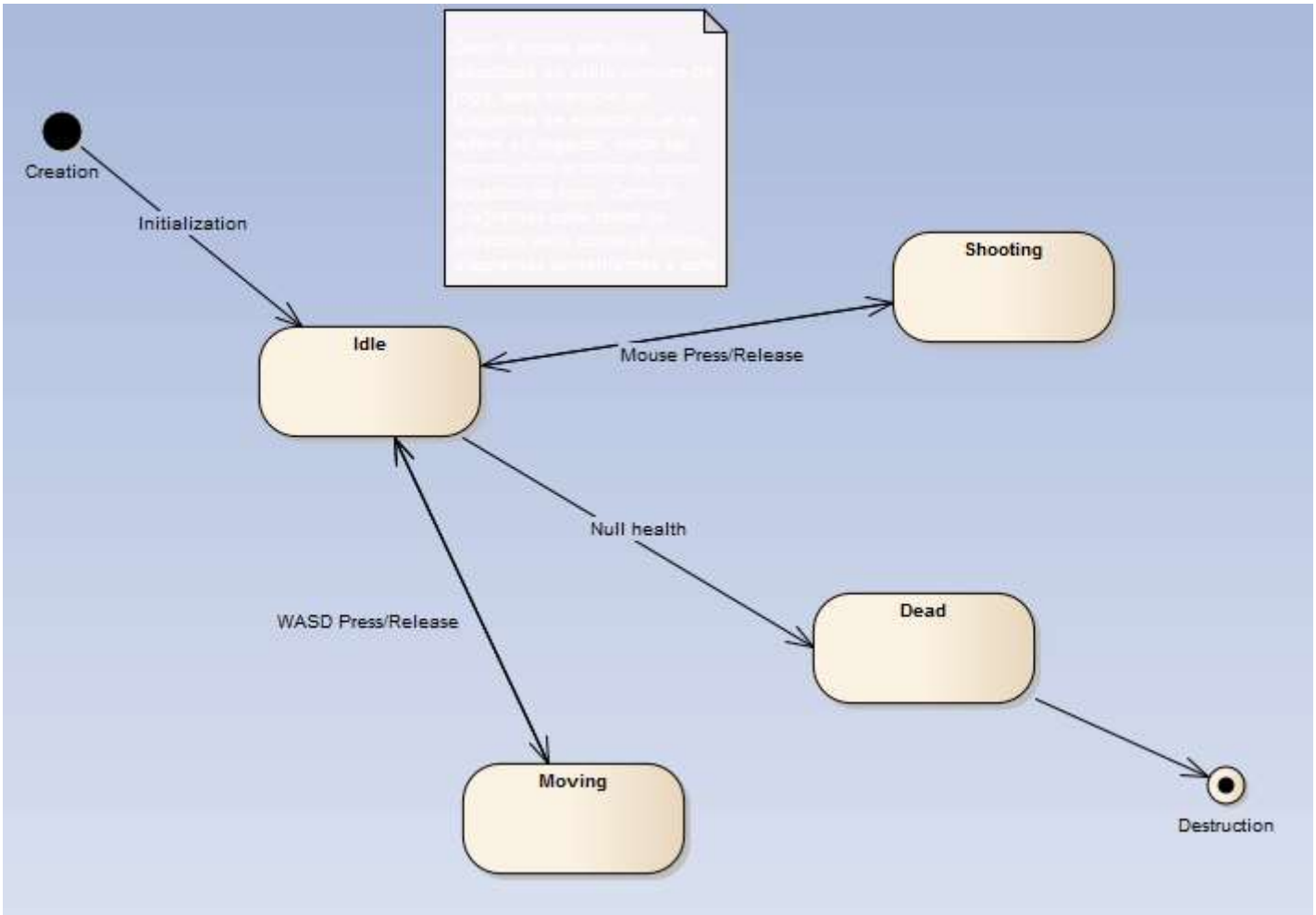
Estrutura de Packages

Os packages encontram-se organizados da forma que achámos mais óbvia. De princípio temos a package responsável pelos objetos presentes no jogo. Em termos do que se sucede no jogo em si, fazemos a divisão entre as classes que tratam de todo o rendering dos objetos no ecrã, e as classes que tratam de toda a lógica necessária ao correto funcionamento do jogo.



Mecanismos Importantes

Um mecanismo importante será o comportamento dos objetos de jogo, neste caso exemplificado pelo comportamento do jogador, que se assemelhará ao comportamento de todos os outros objetos durante o processo do jogo.



Bibliotecas, tecnologias e ferramentas utilizadas

Para o desenvolvimento deste projeto, foram utilizadas algumas ferramentas para além das referidas durante as aulas da Unidade Curricular. O IDE utilizado foi o IntelliJ Idea, por preferência do grupo. Foi, também, usada uma biblioteca externa, *libgdx*, para ajudar no desenvolvimento. Para Sistema de Controlo de Versões, foi utilizada a tecnologia *Git*, mais propriamente um repositório público na página *GitHub*.

Dificuldades encontradas e sua resolução

Uma das dificuldades encontradas, inicialmente, foi o desenvolvimento a nível gráfico. Ao contrário do Dragon Dungeon, este projeto exigia movimentos suaves e contínuos, pelo que foi decidido o uso da biblioteca anteriormente referida, *libgdx*. A deteção de colisões e saídas de objetos da janela de jogo causavam alguns erros de *runtime*, que foram corrigidos com alguma pesquisa e algum estudo.

Conclusão

Grau de Cumprimento dos Objetivos

O grupo considera que os objetivos principais foram alcançados. No entanto, há algumas *features* que gostava que o Kernel Defender tivesse, mas que não puderam ser incluídas. No entanto, num futuro próximo, e pelo entusiasmo com este projeto (que é o primeiro, durante o nosso percurso da faculdade, onde temos liberdade para desenvolvermos algo realmente a nosso gosto para uma Unidade Curricular), pretende-se continuar o seu desenvolvimento (com inclusão das melhorias descritas no ponto seguinte), desta vez para o Sistema Operativo móvel Android, e o lançamento na Play Store, a título de experiência. No geral, há bastante satisfação com o resultado apresentado.

Melhorias Possíveis

- ➔ Inclusão de novas armas (desbloqueáveis e adquiríveis)
- ➔ Possibilidade de *upgrade* das armas
- ➔ Inclusão de *power-up*'s, como por exemplo uma *Firewall* temporária e impenetrável
- ➔ *Port* para Android
- ➔ Implementação de sistema de níveis
- ➔ Implementação de novos modos de jogo (infinito, por exemplo)
- ➔ Possibilidade de customização de opções de jogo
- ➔ Recorrer a técnicas de "pooling", assim evitando que cada sprite carregue a sua textura de memória cada vez que um novo objecto de jogo é criado

Nível de Contribuição dos elementos do grupo

Diogo Miguel Sousa Barroso

- ➔ Programação
- ➔ Arte gráfica
- ➔ Relatório
- ➔ Design e planeamento

Total – 40%

Miguel Geraldes Antunes Mendes

- ➔ Programação
- ➔ Som
- ➔ UML
- ➔ Design e planeamento

Total – 60%

Referências

Libgdx

http://libgdx.badlogicgames.com/

IntelliJ Idea

http://www.jetbrains.com/idea/

Repositório Github

https://github.com/Diogobarroso/LPOO_Kernel_Defender