Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Eletrónica e de Computadores

Caloiros De Elite: *Space Invaders*

Complementos de Programação de Computadores

Ano Letivo de 2021/2022

a101168, Afonso Gomes
a101170, Christian Garcia

Guimarães, maio de 2022

Resumo

Este trabalho consiste numa explicação extensa da criação do jogo **"Caloiros de Elite"**, inspirado no jogo **"Space Invaders"**, desenvolvido pela *Taito Corporation* [1].

Apesar dos seus controlos simples, comparados com os jogos de hoje, este jogo ajudou a expandir a indústria de videojogos para uma indústria mundial. Deste modo, vamos descrever como pensamos e implementamos este programa em linguagem C++ [2] e a biblioteca ALLEGRO5 [3], tendo por base os conceitos teóricos aprendidos nas aulas da unidade curricular.

Através das nossas pesquisas conseguimos encontrar várias formas de contornar os problemas que apareceram, bem como diversas maneiras para os resolver, no entanto, procuramos sempre solucioná-los de uma forma alternativa baseada nos nossos raciocínios. Assim sendo, ao longo deste relatório iremos mostrar, na melhor das nossas capacidades, o pensamento crítico que todos os elementos do grupo tiveram ao longo da implementação do projeto. No fim do mesmo, também se poderá encontrar o código implementado, dividido pelas partes essências do mesmo.

Índice

Índice de figuras

1 Introdução

Grande parte da população que programa as ferramentas de *software* que utilizamos na atualidade viu-se motivada pelos videojogos da geração dos 70s, 80s e 90s. *PACMAN, Tetris* e *Space Invaders* são alguns exemplos deste fenómeno que cada dia influencia mais jovens e adultos a mergulharem no imenso banco de dados que conhecemos como a Internet e começarem a programar.

O objetivo deste trabalho é desenvolver um jogo, inspirado por *Space Invaders*, utilizando as diferentes ferramentas fornecidas pela linguagem de programação C++, nomeadamente, a utilização de OPP (*Object Oriented Programming*), leitura e alteração de ficheiros, entre outros.

O trabalho distribui-se em diferentes temáticas, sendo estas a descrição geral do jogo (natureza e estratégias utilizadas para a sua implementação), arquitetura do sistema (são apresentados, de forma geral, os módulos utilizados e a estrutura do jogo) e finalmente a implementação do mesmo, sendo neste último onde se vai expor ao pormenor cada uma das soluções utilizadas no presente problema.

2 Descrição do Problema

Caloiros de Elite é um jogo do subgénero *Shoot'em Up*, onde o jogador, representado de forma predeterminada pelo símbolo do curso de Eletrónica, tem de acabar com diferentes ondas de inimigos, representados pelos outros cursos de engenharia, e *bosses*, representados pelos departamentos de Medicina e Ciências e pela própria Universidade do Minho.

O jogo tem duas modalidades principais: o modo "história", onde se tem de sobreviver o ataque de diferentes inimigos ao longo de três níveis diferentes, e o modo *"Endless"*, em que procura-se obter a máxima pontuação possível enquanto se sobrevive e acaba com inimigos gerados de forma aleatória, infinitamente.

A primeira modalidade, o modo história, consta de quatro níveis de dificuldade, transformando o jogo numa experiência mais desafiante, sendo o último, "*Hardcore*", o mais difícil deles todos, com só uma vida. Ao longo dos três níveis que conformam esta modalidade, o jogador tem de sobreviver e derrotar aos diferentes *bosses* mencionados anteriormente, acabando por se enfrentar a uma versão final da Universidade de Minho, como inimigo definitivo do jogo. O modo *Endless*, em contraste, permite ao jogador uma experiência mais casual, sem perder a natureza desafiante que forma parte intrínseca do jogo (e da vida universitária). Neste, o jogador enfrenta ondas infinitas de inimigos e *bosses*, na tentativa de sobreviver o máximo tempo possível e acabar com a maior quantidade de inimigos, resultando numa maior pontuação final.

Além da mecânica geral do jogo, este também permite a personalização da nave do jogador, como resultado da realização de certos desafios. Os prémios recompensam maioritariamente a habilidade do jogador e/ou tempo de jogo. Estes dados todos (pontuações máximas, prémios obtidos, estatísticas) ficam salvos no fim de cada experiência e podem ser visualizados em espaços determinados para o efeito (*Hall of Fame*, *Trophies*, *Stats*...)

Para conseguir criar esta experiência de jogo, é necessário recorrer à OOP, sendo que a maioria dos elementos pode e deve ser representada como objetos, com atributos e métodos. Desta forma, conseguimos simplificar o código, iterando funções por entidades semelhantes.

Tendo em consideração a dificuldade do projeto, além das estratégias mencionadas previamente, é necessário a utilização do paradigma *"Divide and Conquer"*, solucionando problemas de menor dimensão e combinando os resultados obtidos para atingir um objetivo final.

3 Arquitetura do Sistema

O programa "Caloiros de Elite" através do ponto de vista do utilizador:

Menu principal: O utilizador escolhe de entre várias opções o que deseja fazer:

- *Play*: opção principal do menu que se subdivide em 3 outras opções:
  - *New Game*: o utilizador opta entre 3 *slots* distintos para guardar o jogo atual. Após a seleção são mostradas 4 dificuldades distintas para o modo história *(easy,normal,hard e hardcore)* nas quais variam a probabilidade dos inimigos dispararem, a vida dos *bosses* e, no modo *hardcore* as vidas visto que nesta dificuldade o jogador apenas tem 1 vida.
  - *Load Game*: o jogador escolhe entre 3 *slots*, previamente inicializados na opção *New Game*, e continua o jogo presente no *slot* selecionado (com a excepção da dificuldade *hardcore*).
  - *Endless* : é pedido ao utilizador para inserir um nome que aparece em tempo real com a fonte utilizada ao longo do trabalho. Em seguida começa o jogo desde o início com 2 *bosses* (um *miniboss* e o *boss* principal que contém uma barra de vida exibida no topo do ecrã). Este modo é infinito sendo que o objetivo é conseguir a maior pontuação possível.
- *Highscores*: são exibidos os *scores* mais altos, em ordem decrescente de cima para baixo, obtidos no modo *endless* e o nome de quem obteve tal pontuação, nome esse que é inserido quando *endless* é inicializado.
- *Trophies*: são mostrados os diferentes troféus adquiridos após a conclusão de certos desafios.
- *Settings*: opção onde o utilizador pode alterar o programa a seu gosto.
  - *Costumization*: são exibidos vários nomes de curso que o utilizador pode optar para trocar a *Skin* atual da nave, o que influencia os inimigos exibidos (não há inimigos iguais à nave).
  - *Technical settings*: a resolução da janela e nível do som podem ser alterados através desta opção.
- *Credits*: permite a visualização dos nomes dos criadores do jogo.

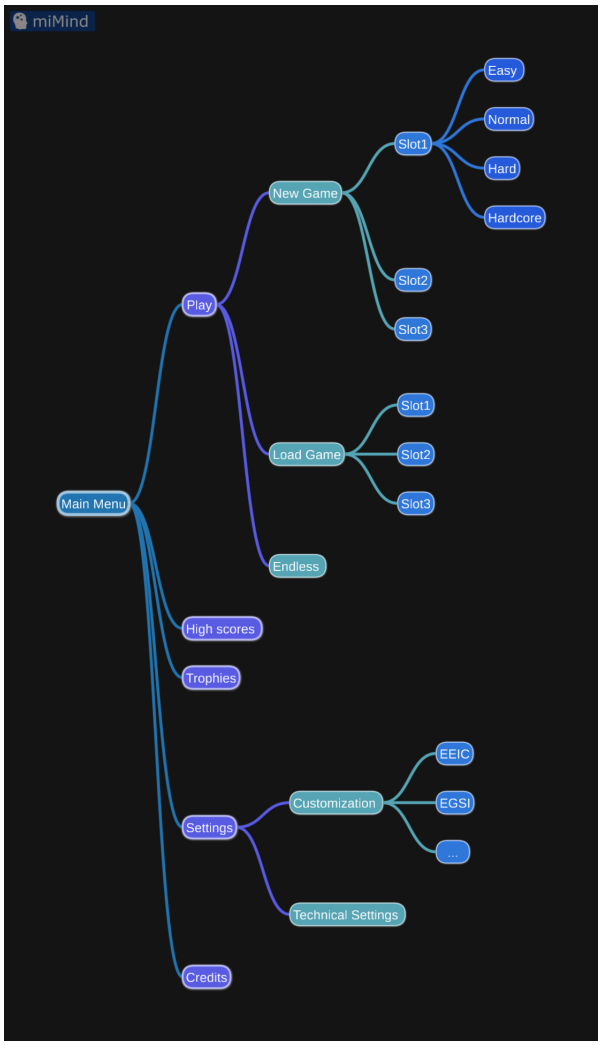

Figura 3.1: Diagrama de blocos da estrutura do jogo desde o ponto de vista do utilizador

O jogo, em nível de estrutura interna, pode ser divido em ”experiência de jogo”, gestão de dados, interface de utilizador e aspetos extras, como apresentado na figura 3.2, sendo que esta organização não representa de forma direta a própria infraestrutura do código, mas os módulos ou elementos lógicos que compõem o todo que conforma o presente trabalho.

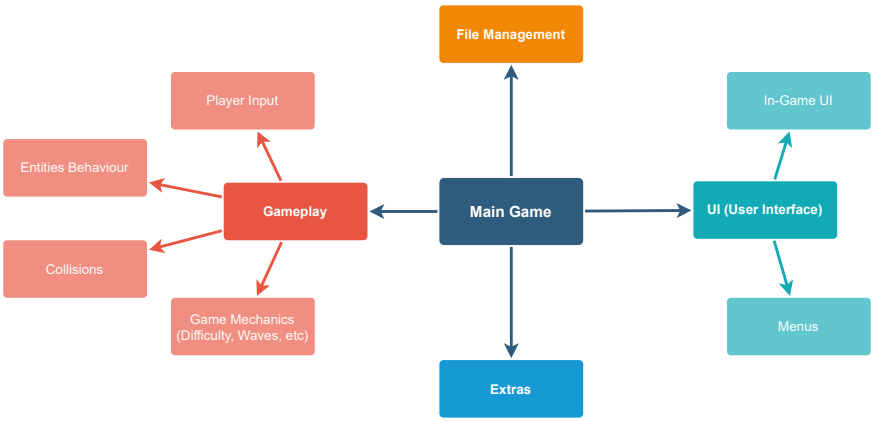

Figura 3.2: Diagrama de blocos no desenvolvimento do jogo

O código só utiliza a linha lógica da estrutura apresentada, sendo que soluciona certos problemas que formam parte de ”membros”do sistema organizacional em classes que representam maioritariamente membros diferentes, com fins de ”simplificar”ou ”reutilizar”código.

Gameplay

O *Gameplay* vê-se refletido na classe *Game*, sendo que esta controla todo aquilo referente aos elementos de uma experiência de jogo (comportamento das diferentes entidades, registo de pontuação se necessário, tratamento de colisões, entre outros elementos designados ao *Gameplay* previamente). Algumas entidades requerem certo grau de complexidade, resultando na sua categorização exclusiva como ”objetos do jogo”, classe *Objects*, engendrando as classes *Player, Bullets, Enemies* e *Bosses* (Figura 3.3). No início de cada experiência de jogo, cria-se um objeto do tipo *Gameplay*, criando este um objeto da classe *Player* e, de forma dinâmica, variedade de objetos das outras subclasses de *Objects*, conforme o jogo se desenvolve.

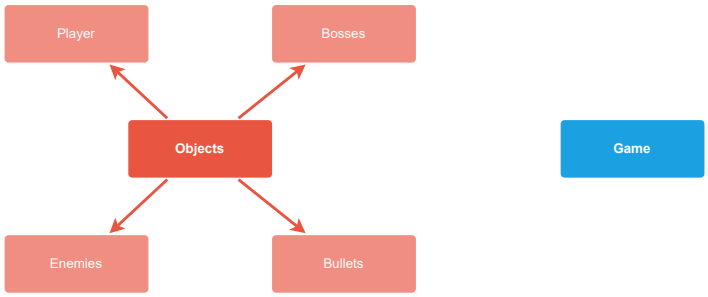

Figura 3.3: Diagrama de blocos da distribuição e estrutura do módulo *Gameplay*

UI - *User Interface*

Embora alguns elementos da UI estão presentes na classe *Game*, nomeadamente, aqueles que têm relação com a *In-Game UI*, a maior parte está representada na classe *Menu*, sendo esta a responsável por gerir e apresentar os *menus* que permitem a visualização de estatísticas e prémios ou a configuração previa a uma experiência de jogo (*skins*, dificuldade e modalidade).

File Management

O *File Management* ocorre maioritariamente no menu onde são carregadas as estatísticas do jogo bem como os *Highscores*, sendo desta forma que se definem os troféus e *Skins* desbloqueadas. Toda a informação previamente mencionada é gravada num ficheiro no final de cada partida (experiêncua de jogo). O *File Management* também aplica na utilização da função *Load Game* onde são carregadas as informações do ficheiro *”save.txt”* aos slots correspondentes. Contudo, quando a dificuldade é *Hardcore* não existe carregamento do jogo atual.

Extras

Os ”extras”do jogo não se encontram confinados num módulo específico, mas sim ao longo dos módulos todos (sendo a classe *Menu*, a principal delas todas). Funções específicas e não essenciais como os *Easter Eggs* (representado pela classe *Bonus*), a possibilidade de escolher naves visualmente diferentes e a realização de troféus acabam por ser elementos que motivam ao utilizador a continuar no jogo.

4 Implementação do Jogo

”Caloiros de Elite”, embora visualmente simplista, resulta diferentes ferramentas informáticas e de código que quando combinados, produzem um jogo desfrutável para o utilizador e satisfatório para os criadores.

Para isto, criaram-se as seguintes classes, que permitem a realização do trabalho:

- *Menu*: Encarrega-se de gerir tudo o relacionado ao menu inicial.
- *Game*: Encarrega-se de gerir a interação dos diferentes elementos de uma experiência do jogo (colisões, movimento ...), simplificando e organizando o código. item *Objects*: Classe base das classes que representam elementos ativos numa experiência de jogo. Simplifica o código, reutilizando membros e métodos comuns.
- *Player*: Classe que permite a criação e controlo do ”jogador”.
- *Enemies*: Classe que permite criar e controlar inimigos.
- *Bosses*: Classe que permite criar e controlar inimigos especiais (*bosses* e *megabosses*).
- *Bonus*: Classe que controla o Easter Egg da modalidade *Endless*.

Além de um ficheiro extra para funções úteis ao longo do programa. A seguir, o .h do mesmo.

```
#include "Objects.h"
#ifndef FUNCS
#define FUNCS
bool verifyCollision(int*, int*); //allows to verify collisions
bool animation(int start, int end, int&, int&, bool ping_pong);
bool compareHighs(const HighscrStat& first, const HighscrStat& second);
void rangeDef(int &val, int* rang); //define range for options
#endif
```

4.1 Representação do Estado do Jogo

No estado mais básico, a aplicação apresenta dois estados: menu e jogo. Estes estados estão definidos pela variável *”playing”*. Os métodos de cada de um dos objetos são realizados com base no estado desta variável. Além disto, cada um destes elementos possui também variáveis que definem o estado delas. No caso do menu, o array ”*int opt[2]*”define o estado e a seleção atual. Estes diferentes estados podem ser visualizados, com o seu respetivo nome no array ”*string options[27]*”.

Quando as escolhas realizadas pelo utilizador resultam na inicialização de uma experiência de jogo, reiniciam-se as variáveis da instância da classe *Game*, adquirindo os valores fornecidos pela instância da classe *Menu* (sendo estas alteradas com base no *input* do utilizador).

Quando o jogo acabar, produto da morte ou da finalização do mesmo (o membro variável *victory* da classe *Game* representa este estado), após salvar os dados necessários, este volta ao estado do menu, reiniciando-o.

4.2 Inicialização do Estado do Jogo

Na inicialização do jogo, o menu abre, carregando a informação necessária dos ficheiros de informação *"save.txt", "highscores.txt"* e *"stats.txt"*, para assim desbloquear os diferentes elementos do jogo. Como exemplo, temos o carregamento dos *slots* do jogo no início do menu.

```
int i = 0;
std::ifstream save_i((".\\save.txt"));
if (save_i.is_open())
{
    while (std::getline(save_i, line))
    {
        if (line.front() == (char)0){
            slots[i][0] = (int)line.front() - 48;
            slots[i][0] = (int)line.back() - 48;
        }
    }
    save_i.close();
}
```

Com base nas opções escolhidas no menu, o jogo inicia, tomando como referências os dados mencionados anteriormente.

Figura 4.1: Ecrã inicial do *Main Menu*

4.3 Visualização do Estado do Jogo

A parte gráfica deste projeto é bastante única visto que todas as entidades foram desenhadas por nós com inspiração nos vários departamentos e cursos de engenharia da Universidade do Minho. Para tal foram utilizadas ferramentas externas. Para dar mais vida ao jogo, além de desenhar as entidades, estas são animadas, recorrendo à utilização de um segundo *Timer*, diferente ao responsável pela maioria das interações e alterações de estado no jogo. Este é responsável por alterar as imagens apresentadas, dando a ideia de que os elementos estão se a mexer.

No caso do jogador e dos inimigos, criamos um *sprite* (Figura 4.2) que contem a visualização os diferentes estados que estes podem possuir, com a sua respetiva animação.

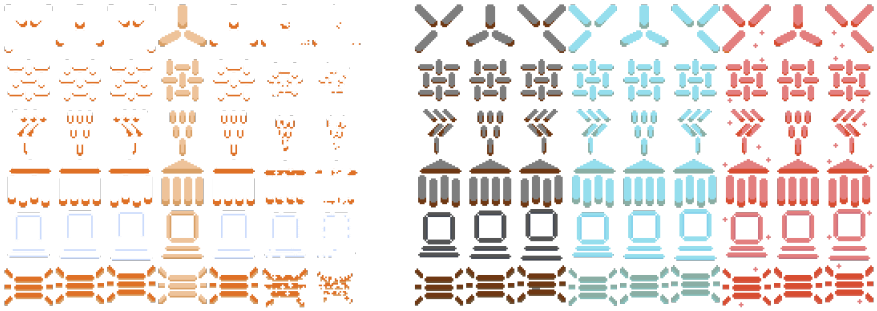

Figura 4.2: *Skins* do jogador/inimigos

Recorrendo às funções do ALLEGRO 5, criamos métodos ".*draw*"nas diferentes classes do jogo. Desta forma, aninhando as funções, podemos simplificar o código, mantendo limpo o código da nossa *main*: *"program.cpp"*. Uma interface típica numa experiência de jogo (Figura 4.3) contém as vidas do jogador, barra de vida do *Megaboss*, se houver, e *score*, no caso de estar a jogar na modalidade *Endless*.

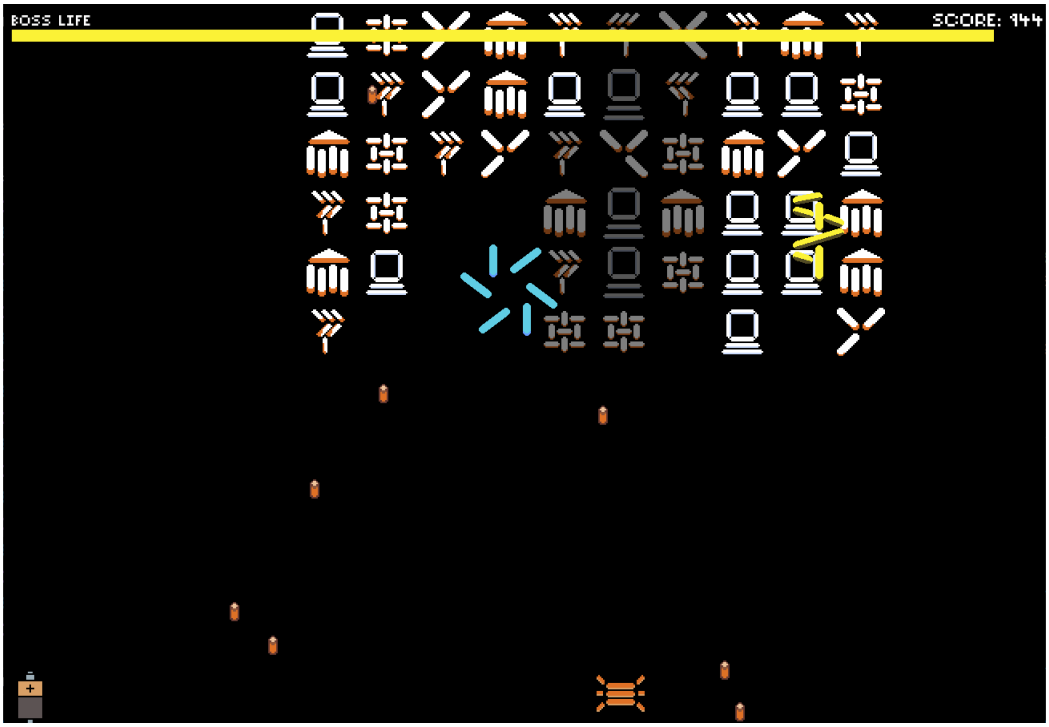

Figura 4.3: Experiência de jogo

4.4 Mudança de estado/movimento do Jogador

O jogador começa com um número de vidas correspondentes à modalidade utilizada (depende de se este for *Hardcore* ou não), que podem aumentar ou diminuir conforme aconteçam certos eventos (atingir uma certa pontuação, destruir *bosses* ou diminuir caso seja atingido). A nave do jogador pode também ter vários estados, obtidos através da destruição de *Bosses* ou de ser atingido pelos mesmos. Existem 6 estados:

- Enfortecido: O dano realizado por cada disparo individual equivale a cinco disparos durante uma quantidade de tempo. Obtém-se acabando com o *boss UM*.
- Protegido: Durante uma quantidade limitada de tempo, qualquer dano recebido não é considerado, retirando esta proteção adicional. Obtém-se quando se ganha uma vida adicional enquanto se está no máximo de vidas possíveis.
- Invulnerável: Parecido a protegido. Não é retirada se o jogador colidir com uma bala. Obtém-se se o jogador receber dano (durante uma quantidade pequena de tempo) ou se se acaba com o *boss Ciências*.
- Lento: O jogador mexe-se a um meio da velocidade normal durante uma quantidade de tempo. Produto de receber o *laser* do *boss Medicina*.
- Rápido: O jogador mexe-se ao dobro da velocidade normal durante uma quantidade de tempo. Obtém-se acabando com o *boss UM*.
- Impedido de Disparar: O jogador não pode disparar durante uma quantidade de tempo. Produto de receber o *laser* do *boss UM*.

Todos os estados anteriores têm uma imagem diferente que serve de indicador para o utilizador poder distinguir do seu estado normal.

O movimento do jogador é bastante simples visto que este apenas pode se movimentar para a esquerda e para a direita, a depender dos eventos do teclado (teclas A/Esquerda e D/Direita, nomeadamente).

4.5 Salvar e Restaurar Jogo

No final de cada experiência de jogo (morte do jogador ou finalização do modo história), os dados (stats, save e highscores, a depender da experiência de jogo atual) são salvos por meio da utilização da biblioteca ≪iostream¿", alterando ou criando documentos ".txt"., com base nos dados obtidos nessa instância e dos presentes nos documentos mencionados previamente. Temos como exemplo, um típico documento de *highscores*.

```
flu 29139
black 21684
homeru 19001
homers 10811
```

Estes dados são carregados cada vez que reinicia o menu, utilizando-os para diferentes fins (carregamento do jogo, troféus, *skins* ...).

Quando um *Slot* é selecionado a partir do submenu *New Game* todos os dados desse espaço irão permanecer no mesmo, com a excepção da dificuldade *Hardcore*. No início de cada nível as estatísticas do utilizador até ao presente momento são guardadas num ficheiro. Quando o jogador morre ou sai do jogo ele pode, através da opção *Load Game*, continuar o modo campanha começando no nível onde deixou.

Figura 4.4: Seleção de dificuldade antes do inicio de um *New Game*

4.6 Final da Execução

Este projeto não tem um fim definido, apenas acaba quando o utilizador fecha a janela do jogo. Sendo assim focar-nos-emos no fim da sessão, ou seja, quando o jogador morre ou conclui o modo história. Quando o utilizador morre no modo história em qualquer dificuldade, o jogo tem em conta vários critérios:

- Número de inimigos mortos;
- Número de *Bosses* eliminados;
- Se o jogador concluiu com sucesso ou não o modo história, e a dificuldade em que estava;

É depois atualizado o ficheiro das estatísticas com estes dados. Se o jogo estiver numa dificuldade diferente de *Hardcore* é também gravado o nível onde o *Player* se encontrava antes de morrer podendo este ser carregado mais tarde.

Caso tenha sido selecionado o modo *Endless* o *Highscore* é guardado os pontos obtidos bem como o nome associado a tal pontuação. É também guardado se houve ou não a destruição de uma nave especial que aparece com uma chance aleatória. Os *Highscores* são automáticamente organizados de forma a apenas aparecerem as 8 melhores pontuações.

Estas informações são mais tarde utilizadas para desbloquear os troféus, exibir os *Highscores* e

atualizar as estatísticas do utilizador.

Figura 4.5: Troféus

Figura 4.6: *Highscores*

5 Conclusões e Perspetivas do Trabalho

Ao longo do trabalho foram-nos surgindo vários desafios e problemas que, com muito empenho e pesquisa, conseguimos resolver. Podemos também ter muita liberdade criativa visto que o tema base não nos restringiu de alguma forma, dando nos assim a liberdade para utilizar outras ferramentas para desenhar os *sprites* de todas as entidades do jogo.

Após a realização deste trabalho, conseguimos concluir que as classes lecionadas ao longo do semestre são realmente importantes na linguagem de C++, pois permitem uma melhor organização do código facilitando o trabalho do programador. Além disto, também foi possível apreciar as vantagens da utilização de programação orientada a objetos. Sem menor importância, também mencionamos os benefícios da utilização de controlo de versões, nomeadamente o Git, sendo que conseguimos trabalhar em conjunto com maior facilidade porque os nossos ficheiros, tanto código [4] como o nosso relatório [5] foram realizados com esta ferramenta.

Por fim, em jeito de conclusão, pode dizer-se que este trabalho foi concluído com sucesso, pois conseguimos focar todos os pontos essenciais para a total conclusão do trabalho. Apesar de concluído com sucesso, é importante referir que ao longo de todo o processo de resolução, principalmente na fase de implementação e modulação, surgiram algumas dúvidas, dúvidas essas que felizmente não foram uma barreira que impediu a concretização deste trabalho. Contudo, houve certas funções que gostaríamos de ter implementado que acabaram por não ser concluídas, nomeadamente as *technical settings*. Esta função não tem influência na execução do jogo, motivo pelo que decidimos não implementar na sua totalidade.

Referências Bibliográficas

[1] Square Enix. Taito - 40 Years of Invading. URL: https://spaceinvaders.square-enix-games.com/news/taito-40-years-of-invading. Visitado por última vez em 16/04/2022.

[2] C++ Docs. URL: https://www.cplusplus.com/reference/.

[3] Allegro Docs. URL: https://www.allegro.cc/manual/5/.

[4] Afonso Gomes Christian Garcia. Caloiros De Elite. URL: https://github.com/Rexrder/CaloirosDeElite. Visitado por última vez em 22/05/2022.

[5] Afonso Gomes Christian Garcia. Caloiros De Elite Docs. URL: https://github.com/Rexrder/CaloirosDeEliteDocs. Visitado por última vez em 22/05/2022.