U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Laboratório de Computadores

2LEIC018 – Turma 14 Grupo 1

2 de junho de 2024

Bruno Aguiar	**up202205619**@up.pt
Francisco Fernandes	**up202208485**@up.pt
Lara Coelho	**up202208689**@up.pt
Paulo Coutinho	**up202205692**@up.pt

Índice

1. Introdução

Este projeto consiste num jogo inspirado em “Lovers in a dangerous spacetime”, mas com muitas alterações.

O jogo tem lugar num oceano, onde um submarino navega. Os jogadores conseguem movimentar-se dentro do submarino e chegar a diferentes sítios onde podem realizar diferentes ações, tais como conduzir o submarino e disparar os seus canhões.

No entanto, o tanque de oxigénio do submarino é limitado e, por isso, é necessário vir à superfície para o renovar. Na superfície, existem inimigos dos quais os jogadores têm de se defender.

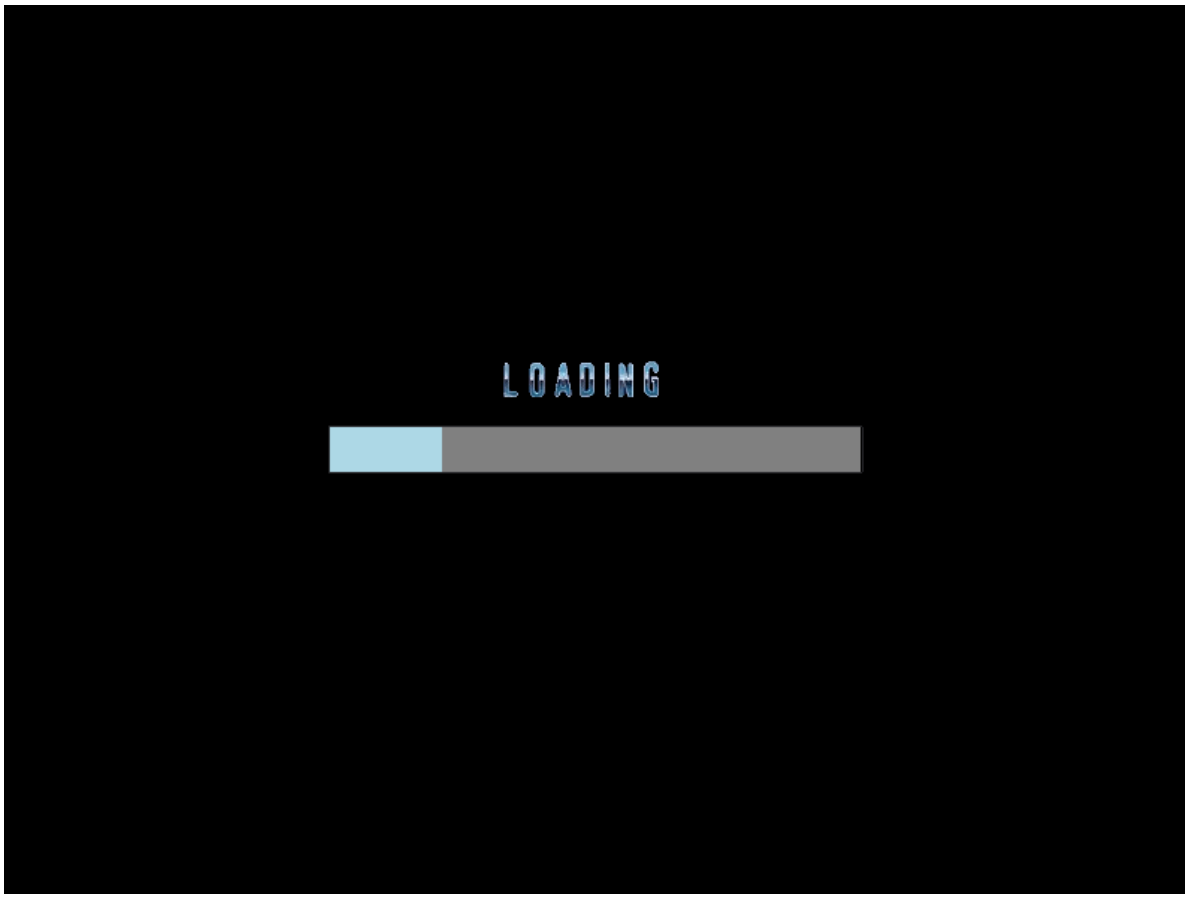
Os jogadores podem apanhar o lixo que encontram em bolhas no oceano, ganhando moedas, para poderem melhorar as suas condições, designadamente reaver vidas perdidas, melhorar o tanque de oxigénio, aumentar a taxa de disparo dos canhões e aumentar a velocidade do jogador.

Quanto mais fundo apanharem o lixo, mais dinheiro ganham. Contudo, se, ao tentar rebentar as bolhas, matarem algum peixe, perderão moedas.

2. Instruções de utilização do programa

2.1 Ecrã inicial

Ao iniciar o programa, é apresentado um ecrã com uma barra que indica o progresso de *load* das *sprites* utilizadas no jogo.



2.2 Menu

Em seguida, o utilizador é redirecionado para o menu onde pode escolher uma das 4 opções.

Play: inicia um novo jogo em modo *single player*.

Multiplayer: inicia um novo jogo em modo *multiplayer.*

Leaderboard: mostra os 5 primeiros melhores resultados obtidos no computador em que o jogo está a ser jogado.

Quit: termina o programa.

2.3 *Singleplayer*

O jogo começa com o submarino debaixo de água, o tanque de oxigénio cheio, 3 vidas e 0 moedas.

À medida que o tempo passa, o oxigénio vai diminuindo e a única forma de reabastecer é emergindo.

As teclas **WASD** tem duas funções: mover o jogador ou mover o submarino se o jogador entrar no modo de conduzir o submarino, dirigindo-se ao leme e pressionando **E**.

No modo de conduzir o submarino, o jogador pode também usar o *scroll* para mover o anzol.

Para utilizar os diversos canhões, o jogador deve dirigir-se à localização desejada e pressionar **E**.

Para sair de qualquer um dos modos, é necessário voltar a pressionar **E**.

Quando chega à superfície, o jogador é atacado por inimigos dos quais se tem que defender e, consoante o dano sofrido, as suas vidas vão diminuindo.

O jogador só pode mover novamente o submarino depois de ter derrotado todos os inimigos da ronda.

Quando o jogador abate um inimigo, as suas moedas aumentam.

Quando o submarino está submerso, se o jogador matar um peixe, irá perder moedas.

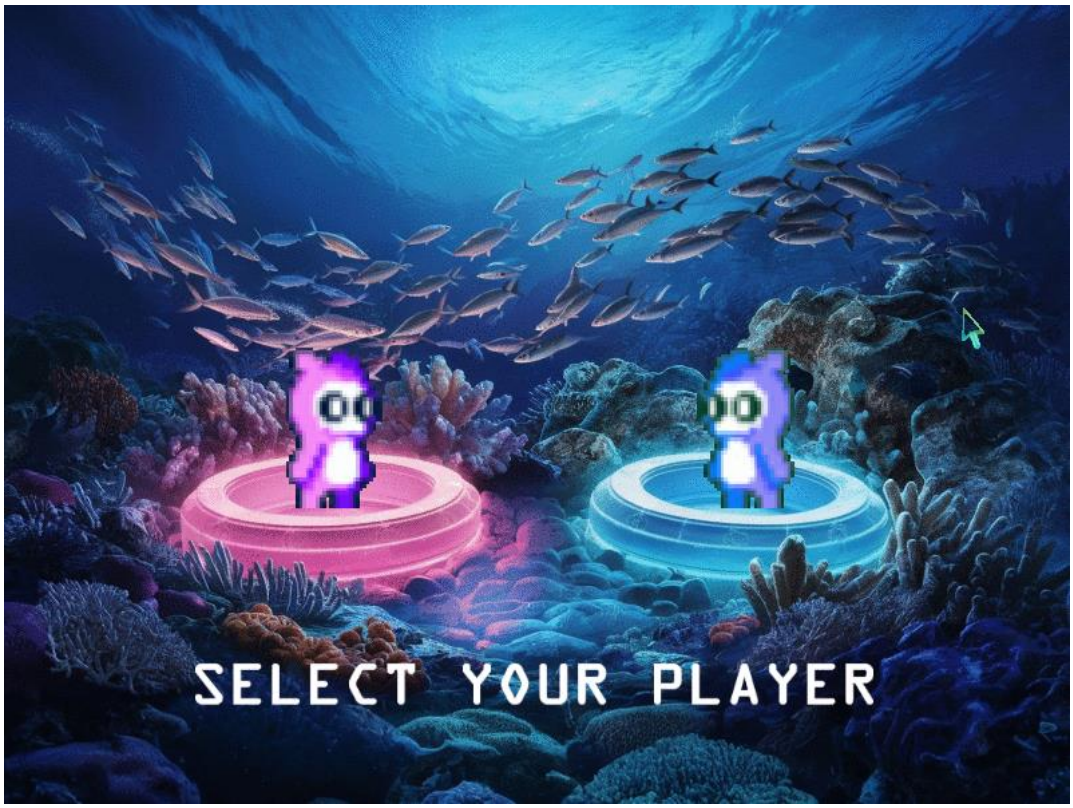
O jogador pode obter recompensas, libertando o lixo das suas bolhas e apanhando-o depois com o anzol do submarino.

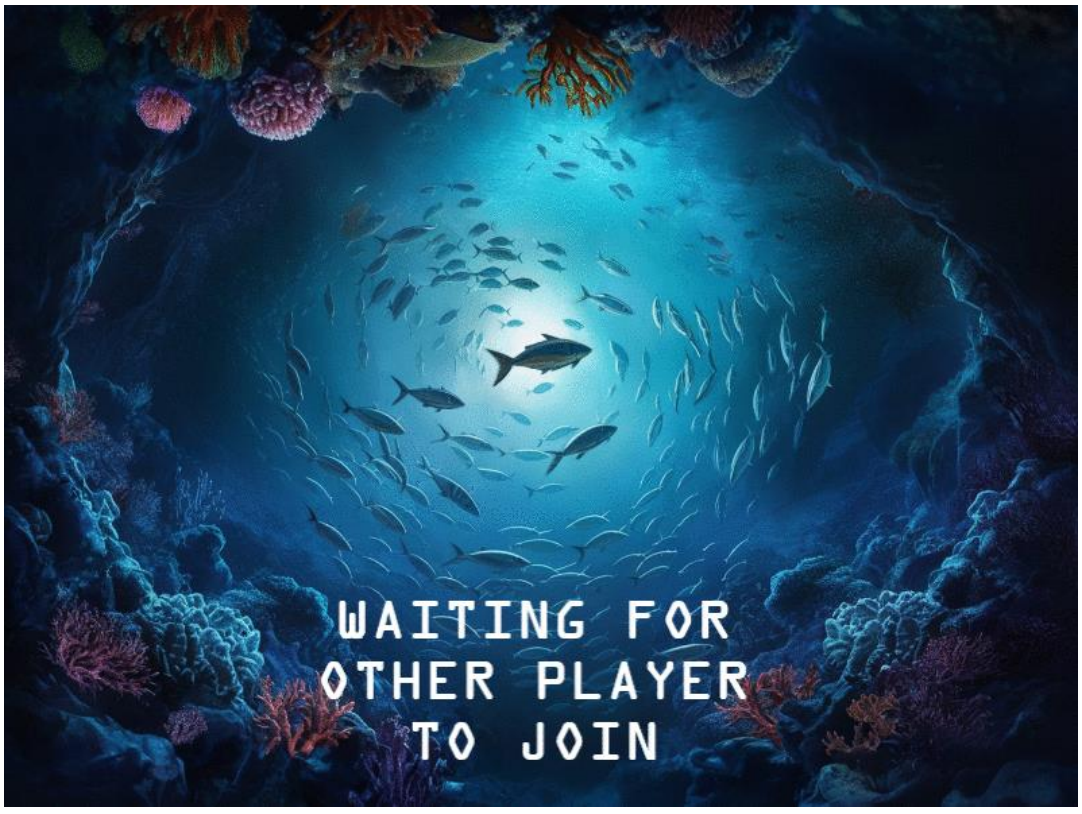
2.4 *Multiplayer*

Cada jogador tem que escolher um jogador distinto para o representar no jogo.

Depois de ambos os jogadores se conectarem, o jogo começa e as funcionalidades são semelhantes ao *singleplayer*.

Neste modo, o objetivo é os jogadores cooperarem de forma a sobreviverem o máximo possível e recolherem o maior número de recompensas.


SELECT YOUR PLAYER


WAITING FOR
OTHER PLAYER
TO JOIN


OXYGEN
30
22:14:15

2.5 *Pause Menu*

A meio de um jogo, é possível fazer pausa, carregando em **ESC**.

Pode-se voltar ao jogo, carregando novamente em **ESC** ou no botão **continue**.

A partir deste menu, é também possível voltar ao menu inicial ou sair do programa.

2.6 Loja

A meio de um jogo, é possível aceder à loja, carregando em **I**.

Pode-se voltar ao jogo, carregando novamente em **I** ou no botão **ESC**.

A partir deste menu, é possível comprar atualizações para a frequência de disparos do canhão, a velocidade do jogador e capacidade de oxigénio. Também é possível recuperar vidas perdidas.

Se o jogador não tiver dinheiro suficiente para comprar um item, o seu preço aparece a vermelho.

Cada item tem um limite de compra. Se este for atingido, aparece um **X**.

2.7 *Leaderboard*

Apresenta os 5 melhores resultados dos jogos que foram jogados no computador com o respetivo nome, *score* e tempo de jogo.

2.8 *Game over*

Num jogo, quando o oxigénio ou as vidas terminam, aparece o ecrã de *game over* e espera *input* do jogador para que este coloque o nome que quer que apareça no leaderboard.

Para ser redirecionado para o *leaderboard,* é necessário pressionar *enter*

3. Estado do projeto

Funcionalidades	Dispositivos	Estado de implementação
Mover jogador	Keyboard e Timer	Totalmente implementado
Mover submarino	Keyboard e Timer	Totalmente implementado
Rotação de canhões e disparo	Mouse	Totalmente implementado
Movimento do anzol para recolha de lixo/ clicar	Mouse	Totalmente implementado
Inimigos	RTC	Totalmente implementado
Criação e mudança de movimento de peixes	RTC	Totalmente implementado
Recolha de texto (nome) do usuário	Keyboard	Totalmente implementado
Mostragem da hora atual	RTC + Placa vídeo	Totalmente implementado
Mostragem das top5 pontuações num leaderboard	RTC + Placa vídeo	Totalmente implementado
Sistema de oxigénio	RTC + Placa vídeo	Totalmente implementado
Comunicação entre dois jogadores	Serial port	Totalmente implementado
Sprite animation		Totalmente implementado
Desenho de sprites com rotação e escala	Placa vídeo	Totalmente implementado
Deteção de colisões		Totalmente implementado
Loja		Totalmente implementado
Cenários de fundo	Placa vídeo	Totalmente implementado

Dispositivo	Funcionalidades	Interrupções
Timer	Controle das FPS do jogo	S
Keyboard	Movimento do jogador e submarino. Entrada/saída dos modos de navegação e canhão. Recolha de texto do usuário.	S
Mouse	Cursor para seleção de opções em menus. Mira de canhões. Descida e içamento e clique do anzol.	S
Graphics Card	Desenho de *sprites.*	N
RTC	Leitura e manutenção da hora atual através de alarmes de 1 segundo. Interrupções periódicas.	S
UART	Comunicação entre dois jogadores.	S

Timer

O *timer* foi utilizado para atualizar a imagem consoante o *frame rate* definido.

Keyboard

O *keyboard* foi utilizado para processar *input* do utilizador, designadamente para movimentar o submarino e os jogadores, para pausar o jogo, escrever o nome para o *leaderboard* e sair do *leaderboard.*

Mouse

O *mouse* foi utilizado para selecionar opções no menu principal e de pausa, para apontar os canhões, disparar e para mover o gancho e apanhar lixo com este.

Configuramos o *Intellimouse* mouse, de modo a transmitir packets de 4 bytes, se for bem sucedido, mas funciona também sem este.

Utilizamos o deslocamento do x, y e z e os botões direito e esquerdo. O botão do meio não foi usado para compatibilidade com *touchpad.*

Graphics card

O *graphics card* foi utilizado para desenhar as *sprites* no ecrã. Utilizamos a resolução 600x800, modo 0x115, com 3 bytes por pixel. Temos *sprites* animadas e colisão entre estas.

Utilizamos *page flipping* para atualização das *frames.*

Temos duas fontes XPM que são usadas para escrever no jogo, na página de game over e no *leaderboard*.

RTC

O RTC foi utilizado para ler a hora atual (atualizando depois através de alarmes a cada segundo), receber interrupções periódicas que atualizam o movimento dos peixes, lixo e inimigos (opção escolhida de forma a sincronizar ambas as máquinas no modo *multiplayer*).

UART

A UART foi utilizada para comunicação entre as duas máquinas no modo *multiplayer*, enviando códigos de conexão, movimentos realizados pelo jogador, pelo submarino, pelo canhão e pelo gancho. Para confirmar a sincronização, também se comunica a destruição de inimigos, de peixes ou de lixo.

4. Organização do código/ Estrutura

1. **Game**
    a. **Game** - Módulo que contém a lógica geral do estado de jogo. Ele coordena vários componentes e garante que o jogo funciona sem problemas. Contém o objeto Submarine, RoundManager, o conjunto de peixes e lixo do jogo e o a entidade que representa o background em movimento. As principais funções são draw_game que chama as funções necessárias para desenhar o jogo atualizado, e as diversas funções que atualizam o seu estado.

    b. **Moving Entities**
        i. **Entity** – A classe base para todas as entidades em movimento. Fornece atributos e métodos comuns para posição, movimento e desenho. Contém funções para a mover e desenhar consoante os diferentes propósitos: move_entity, draw_entity_on_background, draw_entity_on_black_background e draw_entity_on_screen.
        ii. **AnimatedEntity** - Estende Entity para permitir animações, lidando com as atualizações de quadros de *sprites*, sendo a sua principal função animate_entity.
        iii. **Fish** – Estrutura que representa as entidades Peixe do jogo, contém uma AnimatedEntity de um peixe a nadar para uma melhor experiência gráfica do jogador. Se atingidos, morrem e constituem uma penalidade monetária para o jogador. A sua principal função change_fish_direction, altera de forma aleatória a direção dos peixes.
        iv. **Trash** – Estrutura que representa as entidades Lixo do jogo. Trata-se de uma AnimatedEntity com 3 estados distintos: IN_BUBBLE (quando este se encontra dentro de uma bolha de ar, atingível pelo jogador), POPPING_BUBBLE (estado de transição, alcançável ao atingir o lixo na bolha, que permite uma animação de destruição da bolha) e OUT_BUBBLE (estado fora da bolha, o lixo lentamente cai). É no estado OUT_BUBBLE que o jogador pode tentar apanhar o lixo, clicando neste com o anzol do submarino. A sua principal função é trash_pop_bubble, que altera o estado do lixo quando a sua bolha rebenta.
        v. **MainPlayer** - A principal entidade controlada pelo jogador, com comportamentos e interações específicas dependendo do seu estado. Contem uma AnimatedEntity com animação própria

consoante o seu estado. Este pode ser WALK (jogador a mover-se no submarino), DRIVE (jogador a conduzir o submarino), INCANNON1 – INCANNON5 (jogador a utilizar um dos 5 canhões do submarino). Recorre-se à funcão transition_to_state para ir para um estado específico. A função move_main_player move o jogador consoante o movimento e velocidade especificados.

c. **Bullet** – Representa as balas disparadas dentro do jogo, com lógica para movimento e colisões. Quando esta atinge um objeto, esta é removida e cria-se uma explosão, AnimatedEntity temporária, no seu lugar. A função check_bullet_collision verifica se houve uma colisão entre a bala e um objeto e a função explode_bullet explode a bala em caso de colisão.
d. **Cannon** – Gere uma Entity canhão que dispara balas, este é responsável por um conjunto de balas que são ativadas ou desativadas e pelas suas correspondentes explosões. A sua função principal é fire e permite disparar consoante o *fire rate* dado.
e. **Submarine** - Representa o submarino do jogador, combinando movimento e interações com outras entidades. Tem estados SUBMERGED e SURFACED. Contém funções como check_collision que verifica se o jogador se pode deslocar e draw_submarine_and_childs que desenha o submarino e os objetos que este contém, atualizando o seu movimento.
f. **RoundManager** – Estrutura responsável por controlar os Inimigos, suas balas e explosões. Garante o aumento progressivo da dificuldade das rondas. Algumas funções principais como update_round e update_enemies atualizam a ronda de inimigos.
g. **Shop** – Estrutura com items que se pode clicar para comprar, estes têm uma funcão que executa quando é comprado (aumentar velocidade do jogador, capacidade do oxigenio, reposição de vidas e taxa de disparo de canhão).
h. **Hook** – Estrutura que representa o anzol do submarino.
i. **Handler** - Módulo de funções que modificam o Game, consoante o input recebido por serial port, existem diversas funções que atualizam o estado do Game. Tais como: connect_handler, player_handler, sub_handler, cannon_handler, hook_handler, shop_handler e destroy_handler

2. **Devices**
    a. **Timer** - Define as funções timer_subscribe_int e timer_ih para interação com o timer.
    b. **KBC** – Define uma interface para interagir com o KBC, nomeadamente para ler output e escrever comandos nos seus registos.

c. **Keyboard** – Define as funções para interação com o teclado, como keyboard_subscribe_int e kbd_ih que lê o scancode do KBC output buffer.
d. **Mouse** – Define as funções para interação com o rato, como assemble_packet para formar os packets transmitidos pelo rato, e mouse_enable_scroll para ativar a extensão *intellimouse* do rato.
e. **RTC** – Define funções para interação com o Real-Time clock como rtc_get_date para leitura da data atual, e rtc_subscribe_periodic_interrupts para subscrição de interrupções períodicas que ditam o ritmo da lógica do jogo.
f. **SerialPort** - Define funções para interação com a porta série. Algumas das mais importantes incluem: sp_write_wq e sp_read_rq para escrita e leitura das filas de transmissão de dados, e sp_read_fifo e sp_write_fifo para leitura e escrita do buffer FIFO.
g. **VideoCard** – Define funções para interação com a placa gráfica. Implementou-se desenho de *sprites* com *page flipping* para uma melhor experiência gráfica com poucos defeitos visuais. Algumas das funções mais importantes incluem vg_page_flip, vg_initialize e vg_draw_pixel.

3. **Queue** – Estrutura de dados que representa uma fila, usada como apoio ao serial port para leitura e escrita de códigos.
4. **Assets** – Modulo de recursos com todas as XPM utilizadas no jogo.
5. **Utils** – Funções de apoio ao desenvolvimento do projeto como util_get_LSB e util_sys_inb.
6. **Menu**
    a. **Leaderboard** – Define funções para leitura e escrita do nome, pontuação e data do top 5 melhores jogos para um ficheiro.

5. Detalhes de implementação

Utilizamos uma *sprite* com uma *mask* colorida do submarino, assinalando a diferentes cores o chão, as escadas, as paredes, o leme do submarino e os canhões, para facilmente detetar as colisões dos jogadores dentro do submarino.

Assim, o jogador apenas pode andar para os lados se colidir com o vermelho (o chão) e apenas pode subir ou descer se colidir com o azul (as escadas).

Implementamos deteção de colisões entre *sprites:* primeiro, verifica-se se os retângulos das *sprites* coincidem e, se for o caso, para maior precisão, verifica-se se há sobreposição de pixéis com cor entre eles.

O jogo tem os seguintes estados que mudam consoante os eventos: menu principal, jogo em si, menu de pausa, ecrã de *game over*, ecrã de selecionar o jogador para *multiplayer*, ecrã de espera de conexão do outro jogador e *leaderboard.*

Todas as nossas *sprites* podem ser desenhadas com uma dada escala, permitindo facilmente reduzir ou aumentar o tamanho com que a *sprite* é desenhada. Além disso, é também possível desenhar as *sprites* com uma certa rotação, método utlizado para desenhar o submarino e tudo aquilo que a ele está associado. Foi necessário também rodar os canhões, tendo em conta não só a rotação do submarino como também a localização do rato para o qual aponta.

Os inimigos, os peixes e o lixo são gerados tendo em conta uma *seed* que dita todos os seus movimentos, de forma a obter um jogo igual para ambos os jogadores em *multiplayer*.

O lixo tem dois estados - dentro e fora da bolha – e apenas pode ser apanhado depois da bolha ter sido destruída por uma bala disparada pelo canhão.

O nosso *background* faz *loop,* dando a ideia de que é infinito, melhorando a experiência do jogador. Tivemos que ter em conta a posição atual do *background* para desenhar na posição correta os peixes, o lixo e as balas.

A dificuldade das rondas aumenta progressivamente (o número de inimigos e a frequência de disparos).

Quanto maior for a profundidade que o jogador conseguir atingir, maiores são as recompensas.

Temos duas fontes: uma personalizada com letras em tons de azul e com números amarelos, e outra que pode ser desenhada com qualquer cor especificada, para situações gerais.

6. Conclusões

A porta série foi o que demorou mais tempo a integrar. A nossa intenção era usar “state-machine replication”, cada processo mantém o estado complete do jogo, gera localmente as frames a desenhar e processa os inputs tanto do jogador local como do remoto. Isto exigiu diversas tentativas de formas diferentes de sincronização, tendo sido os interrupts periódicos do RTC a nossa escolha final.

Conseguimos implementar todas as funcionalidades que pensamos no início do projeto. E adicionamos algumas que foram surgindo e faziam sentido.

Existem sempre aspetos a melhorar, e nunca ficamos completamente satisfeitos com o produto final, mas estamos orgulhosos do nosso trabalho.

Apêndice: Instruções de instalação

É necessário ter a porta série ativa para correr o jogo.

NOTA: É possível ver os gráficos de chamadas de funções no Doxygen.