Faculdade de Engenharia da Universidade do Porto

Dual Duel: Final Report

Laboratório de Computadores 20/21

Afonso Maria Rebordão Caiado de Sousa **up201806789**

Tomás Freitas Gonçalves **up201806763**

Índice

Instruções de utilização:

Dual Duel é um jogo multijogador local onde os dois jogadores devem disparar sobre o seu adversário, evitando, ao mesmo tempo, as balas do mesmo.

Menu

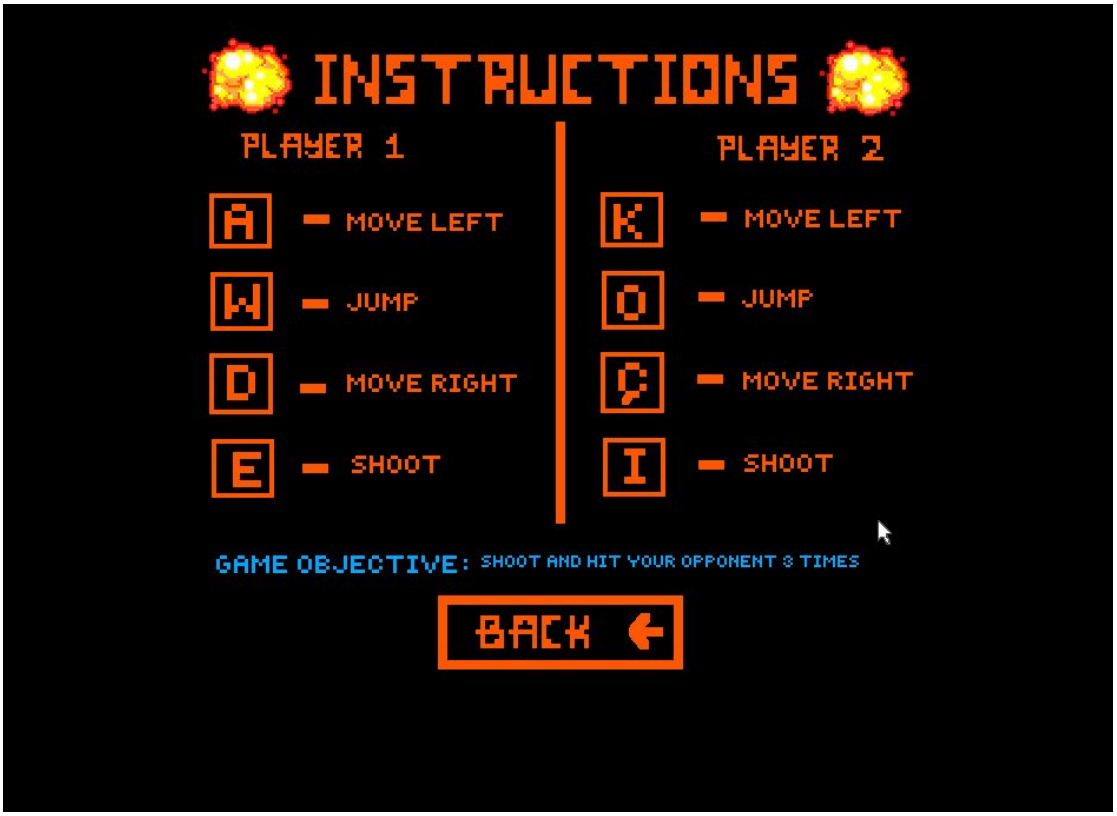
Quando o programa é iniciado é-nos mostrado um menu com as seguintes três opções:

- Play: Inicia o jogo.
- Instructions: Mostra uma janela com as instruções e comandos de jogo.
- Exit: Sai do jogo.

Basta selecionar a opção desejada com a ajuda do rato.

Instruções

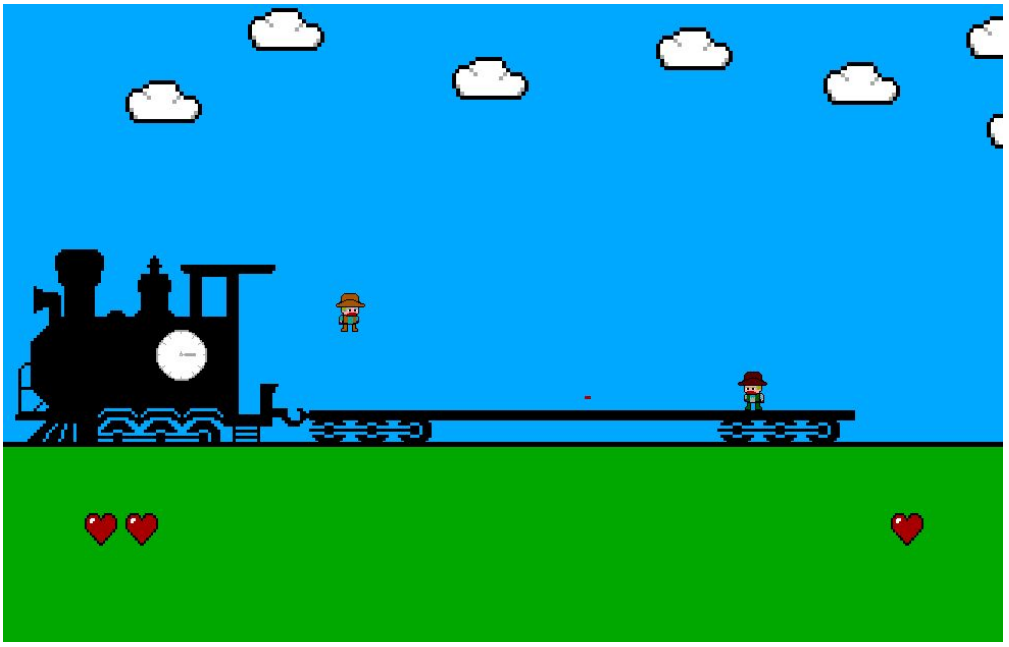
Quando se seleciona a opção das instruções é demonstrado uma lista com as instruções e comandos para cada um dos jogadores.



Jogo

Quando o jogo começa, os dois jogadores estão posicionados sobre uma plataforma de um comboio e podem deslocar-se para a esquerda e direita assim como saltar.

Cada jogador pode utilizar a sua pistola para disparar um tiro horizontal, podendo ser combinado com a funcionalidade de salto para disparar a alturas diferentes. No início do jogo cada jogador possui três vidas que se vão esgotando à medida que os tiros do adversário o atingem. Cada jogador só pode utilizar um tiro de cada vez, sendo obrigado a esperar que este atinja o seu adversário ou saia da zona de jogo para poder disparar novamente. O jogo termina quando um dos jogadores esgota as suas três vidas.



Como indicado no ecrã de instruções, os jogadores devem utilizar o teclado para controlar a sua personagem.

Game Over

Quando o jogo chega ao fim, é exibido um ecrã que indica o jogador vencedor.

Estado do Projeto

Dispositivos Usados

Dispositivo	Utilização	Interrupção
Timer	Atualização de todas as informações do jogo. Nos menus, para reconhecer quando o utilizador clicou no rato, durante o jogo, para atualizar as imagens apresentadas pela placa gráfica.	Sim
Teclado	Interface com o jogo, reconhecendo as teclas para efetuar as funcionalidades dos cowboys.	Sim
Rato	Navegação nos menus, menu principal e menu de instruções.	Sim
Placa Gráfica	Desenho de imagens, nos menus e no jogo.	Não
RTC	Atualização do relógio presente durante o jogo, diferença no fundo do jogo consoante a hora (noite ou dia)	Não

Timer

A função principal do dispositivo I/O Timer é a atualização do estado e das informações do jogo, sobretudo para desenhar os gráficos do jogo. São utilizadas interrupções do rato de duas formas diferentes.

Nos menus (*game.c - create_menu()*), quando é recebida uma interrupção do timer, é revisto o estado do rato.

Durante o jogo (*game.c - start_game()*), quando é recebida uma interrupção do timer, são chamadas 4 funções diferentes, que servem para atualizar os objetos que se movimentam durante o jogo (2 jogadores, as nuvens, ou os tiros disparados pelos jogadores), se necessário.

Teclado

No nosso jogo, o teclado é utilizado para controlo do jogo em si.

Mais uma vez com ajuda de interrupções (*game.c - start_game()*), o nosso programa reconhece os scancodes lidos e efetua ações com estes. Quando os utilizadores pressionam as teclas direcionadas para tal, cada jogador move o seu personagem.

No caso, é possível saltar (jogador 1 - tecla W, jogador 2 - tecla O), mover para a esquerda (jogador 1 - tecla A, jogador 2 - tecla Ç), mover para a direita (jogador 1 - tecla D, jogador 2 - tecla Ç) e finalmente disparar (jogador 1 - tecla E, jogador 2 - tecla I)

Rato

O rato é um dispositivo muito importante para o nosso programa, pois apenas com ele é possível navegar nos menus.

Utilizamos a posição do rato, como também os seus botões, mais uma vez com a ajuda de interrupções.

Para as posições, usamos uma struct Mouse (*game.h*) que guarda as coordenadas x e y atuais do rato. Estas coordenadas são atualizadas a cada interrupção do mouse.

Quanto aos botões do rato, essa mesma *struct Mouse* (*game.h*) guarda também um booleano *clicked* que determina se o botão esquerdo do rato foi pressionado. Informação esta que também é atualizada quando é detectada uma interrupção do rato. De seguida, e com o auxílio das interrupções do timer, quando estas são recebidas nos menus, é analisada a posição (x e y) e o booleano *clicked* para verificar se o utilizador clicou em algumas das opções do menu (play, instructions e exit, ou até back no caso do menu de instruções).

Placa Gráfica

A placa gráfica é o dispositivo base de todo o nosso jogo.

O modo que decidimos utilizar foi o modo *0x105*, iniciado e terminado na função do proj.c, para apresentar imagens com endereçamento indexado de cor (com 8bits por pixel)

Para podermos ter um jogo fluído, implementamos a técnica de *double buffering* nas função *create_menu()* e *display()* do módulo game.c. Onde as imagens são representadas num buffer secundário e mais tarde copiadas para o buffer principal de modo a evitar *flickers.*

Temos vários objetos com movimentação, as nuvens e os tiros que movem de forma periódica e os personagens, que se movem com os cliques nas teclas destinadas para tal.

É detectada a colisão entre os tiros disparados e o personagem adversário, fazendo o personagem que sofre a colisão perder uma vida das 3 que tem disponível à partida.

São desenhados xpms no menu principal (*menu.h*), menu de instruções (*instructions.h*), durante o jogo (todos os restantes ficheiros presentes na pasta imagens), e para apresentar o vencedor (*player1wins.h e player2wins.h*).

Real Time Clock (RTC)

No nosso programa, o RTC é utilizado para ler as horas, minutos e segundos aquando do início da execução do jogo (*rtc.c - get_date()*).

Com esta informação, o nosso jogo muda a cor do background quando é noite (entre as 19h e as 5h) e mostra também um relógio que aponta a hora atual que pode ser visível durante a totalidade do jogo.

Consultar a função *start_game()* no *game.c* e o booleano *night* utilizado.

Organização e Estrutura de Código

Game

Game é o módulo principal do jogo, este contém as funções e guarda as informações do jogo.

A classe tem 14 funções:

-**create_menu()**: Esta função é responsável pela criação e gestão principal do menu. A função começa por subscrever o timer e o rato que serão utilizados no menu principal. De seguida desenha o *xpm* correspondente ao menu.

O ciclo de interrupções do rato faz *parse* dos pacotes e atualiza a *struct mouse* que contém as coordenadas absolutas do ponteiro e um valor booleano correspondente ao clique do botão esquerdo do rato.

Dentro do mesmo ciclo de interrupções é feita a atualização do xpm do ponteiro do rato. Apagando o ponteiro anterior, tapando as coordenadas antigas com a parte correspondente do segundo buffer que contém a imagem de fundo e, de seguida, desenhando o ponteiro nas novas coordenadas *mouse.x,mouse.y*. Esta atualização do rato é feita apenas a cada dois ciclos, para evitar o atraso resultante das constantes atualizações da imagem.

Dentro do ciclo de interrupções do timer é verificado se o botão esquerdo do rato é ativado dentro de alguma zona correspondente a um dos botões do menu principal (*play, instructions ou exit*) ou ao botão de retroceder dentro do menu de instruções.

Finalmente, é feito o unsubscribe dos periféricos.

-start_game(): A função **start_game()** é chamada quando o utilizador começa o jogo. Esta é responsável pelo funcionamento principal do mesmo. A função começa por verificar se o jogo decorre entre as 19:00 e as 6:00 de forma a ativar o modo noturno. Este permite alterar as cores do cenário para um ambiente mais escuro.

De seguida é chamada a função **setup()** que permite atribuir os valores iniciais às variáveis que contêm as informações do jogo.

Mais tarde, é feita a subscrição do timer e do teclado, essenciais nesta parte do jogo. O loop de interrupções decorre enquanto as flags *p1.alive e p2.alive* forem verdadeiras ou enquanto a tecla *esc* não for premida.

O loop de interrupções chama a função **display()**, responsável por representar as cores

de fundo (dependendo da hora do dia), os elementos estáticos (relva e comboio), desenhar o número de vidas restantes (**draw_hearts()**) e desenhar as balas.

As interrupções do teclado permitem reconhecer as teclas essenciais para o controlar os jogadores e ativam *flags* que permitem indicar se o jogador está em movimento para a direita, esquerda, a saltar, disparar *etc*…

Estas flags serão, de seguida interpretadas pelas interrupções do timer que chama as funções:

-**p1Verifications()** e **p2Verifications()**: Responsáveis por atualizar as coordenadas de ambos os jogadores quando as flags de deslocamento ou salto estão ativas.

-**shootingVerifications()**: Permite acionar a animação de disparo, e mantê-la por 50 ciclos de relógio.

-**cloudVerification()**: que altera em permanência a posição das nuvens, deslocando-as para a direita e retornando-as à posição 0 quando atingem o limite do ecrã.

Finalmente, é chamada a função **showWinner()** que imprime no ecrã o jogador vencedor (*Player 1 ou Player 2*) e é feito o unsubscribe dos periféricos usados.

18042

Utilizado o ficheiro fornecido nas aulas com algumas modificações suplementares.

18254

Utilizado o ficheiro fornecido nas aulas com algumas modificações suplementares.

Keyboard

Foram utilizadas as funções do ficheiro desenvolvido durante as aulas práticas.

Mouse

Foram utilizadas as funções do ficheiro desenvolvido durante as aulas práticas.

Proj

É utilizado para iniciar e terminar o modo de vídeo, assim como chamar a função **draw_menu()** e libertar a memória de vídeo no final da execução**.**

Rtc

Apesar de não serem utilizadas, fizemos funções para tratar de subscrever o rtc.

A função principal deste módulo é a função get_date() e as suas derivadas, que serve para ler as horas, minutos e segundos do rtc.

A função final treatHour() é uma função auxiliar à nossa funcionalidade de mostrar um relógio durante o jogo, função esta que é responsável por converter a hora lida numa hora inferior a 12, para ser possível mostrar no relógio.

Timer

Foram utilizadas as funções do ficheiro desenvolvido durante as aulas práticas.

Utils

Foram utilizadas as funções do ficheiro desenvolvido durante as aulas práticas.

Videocard

Foram aproveitadas as funções do ficheiro desenvolvido durante as aulas práticas e foram acrescentadas funções de forma a permitir que fosse efetuado um double buffering. Foi adicionado um novo parâmetro a todas as funções aproveitadas, de forma a que seja indicado em qual dos buffers os elementos devem ser representados.

Detalhes de Implementação

Achamos importante mencionar três estruturas que foram utilizadas para guardar informações sobre o jogo:

```
struct Player{
  int x;
  int y;
  int lives;
  bool alive;
  bool left;
  bool right;
  bool jumping;
  bool falling;
  bool shooting;
};
```

Player: Permite guardar as informações sobre cada um dos jogadores. Guarda as coordenadas do jogador, o número de vidas restantes e flags que indicam se o jogador está vivo, se se desloca para a esquerda ou direita, se está a saltar, a cair ou a disparar.

```
struct Bullet{
  bool visible;
  int x;
  int y;
};
```

Bullet: indica se a bala está visível, tal como as coordenadas da mesma.

```
struct Mouse{
  bool clicked;
  int x;
  int y;
};
```

Mouse: Apresenta as coordenadas atuais do ponteiro do rato e uma flag que indica se o botão esquerdo foi premido.

O conhecimento das coordenadas dos elementos foram essenciais para permitir detetar contacto entre os mesmos. Pensamos, naturalmente, nas colisões entre os jogadores e as balas adversárias ou mesmo do ponteiro do rato com os botões dos menus.

Conclusão

De forma global, o grupo ficou satisfeito com o trabalho realizado, acabando por superar em diversos aspetos a ideia projetada no início do desenvolvimento do projeto. Confirmou-se a utilização dos periféricos planeados (Timer, Mouse, Keyboard, Video Card , RTC), embora se possa tornar, por vezes, complicada a gestão de tantos elementos em simultâneo.

Se devesse ser feita uma crítica ao trabalho final, apesar do número grande de horas dedicadas ao projeto, seria a organização do código, que poderia ter sido mais aprofundada tendo o grupo tido mais tempo.

Consideramos, no entanto, que o grupo desenvolveu um bom trabalho com os conhecimentos adquiridos nas aulas teóricas e laboratoriais, com ambos os elementos do grupo tendo-se empenhado e dedicado ao projeto de forma justa.

Apêndice

Para correr o nosso jogo basta, através do minix aceder ao diretório de raiz do código utilizando o comando:

"**cd .../proj/src"**

De seguida basta correr o comando:

"**make"**

que permite compilar o código e finalmente:

"lcom_run proj"

para correr.