TRON

Laboratório de Computadores

2016/2017

João Pedro Teixeira Pereira de Sá
(up201506252)

Francisco Maria Fernandes Machado Santos
(up201607928)

Índice

1 . Introdução

O tema do trabalho sobre o qual nós decidimos fazer é o jogo Tron.

O jogo tem uma vertente multiplayer e outra singleplayer, cujo objetivo consiste numa corrida de motos onde o primeiro jogador a embater numa fronteira perde.

Este jogo pareceu-nos ser o ideal para realizar uma vez que, vai ao encontro de todo o objetivo da unidade curricular de LCOM, desenvolver código de baixo nível em linguagem C, usar a linguagem C de modo estruturado e aprender a utilizar a interface de “hardware” dos periféricos mais usuais do computador.

2. Manual do Jogo

2.1 Menu Inicial

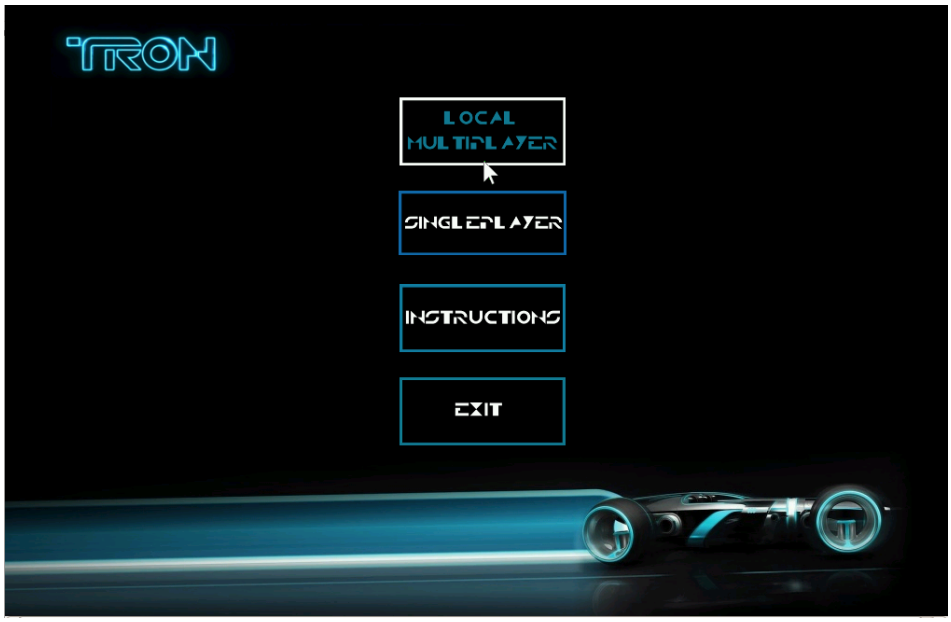


Fig. 1 – Menu Inicial do Jogo

O menu inicial é composto por uma imagem de fundo, pelo cursor do rato e por quatro botões selecionáveis.

O menu é o ponto de partida para qualquer função do jogo, sendo que a seleção de qualquer um dos quatro butões leva à respetiva função.

- **Botões:**

Os quatro butões possuem dois estados, o estado normal e o estado selecionado.

Estado Normal: Apresenta as letras de cor branca e o retângulo envolvente de cor azul.

Estado Selecionado: Apresenta as letras de cor azul e o retângulo envolvente de cor branca.

➔ **Local Multiplayer :**

Este botão, quando premido, inicia um jogo multiplayer.

Fig 1.a) Estado normal

Fig 1.b) Estado Selecionado

➔ **Singleplayer :**

Este butão quando premido inicia um jogo singleplayer.

Fig 2.a) Estado Normal

Fig. 2.b) Estado Selecionado

➔ **Instructions :**

Este butão, quando premido mostra as instruções sobre como jogar o Tron.

Fig 3.a) Estado Normal

Fig 3.b) Estado Selecionado

➔ **Exit :**

Este butão, quando premido, sai do programa.

Fig 4.a) Estado Normal

Fig 4. b) Estado Selecionado

2.2. LOCAL MULTIPLAYER

Neste modo de jogo permite-se um jogo multiplayer a dois jogadores, Player 1 e Player 2. O objetivo consiste em levar o adversário a embater contra uma fronteira ou contra um dos rastos dos jogadores.

Player 1
(e seu respetivo rasto)

Player 2
(e seu respetivo rasto)

De início, ocorre uma contagem decrescente de três segundos de forma a preparar os jogadores para o começo do jogo. Assim que a contagem acabe, o jogo começa e os bonecos involuntariamente começam a movimentar-se, um em direção ao outro, sendo que o seu controlo é apenas possível com as teclas-guia de cada jogador. O primeiro a colidir com qualquer fronteira/rasto perde o "round".

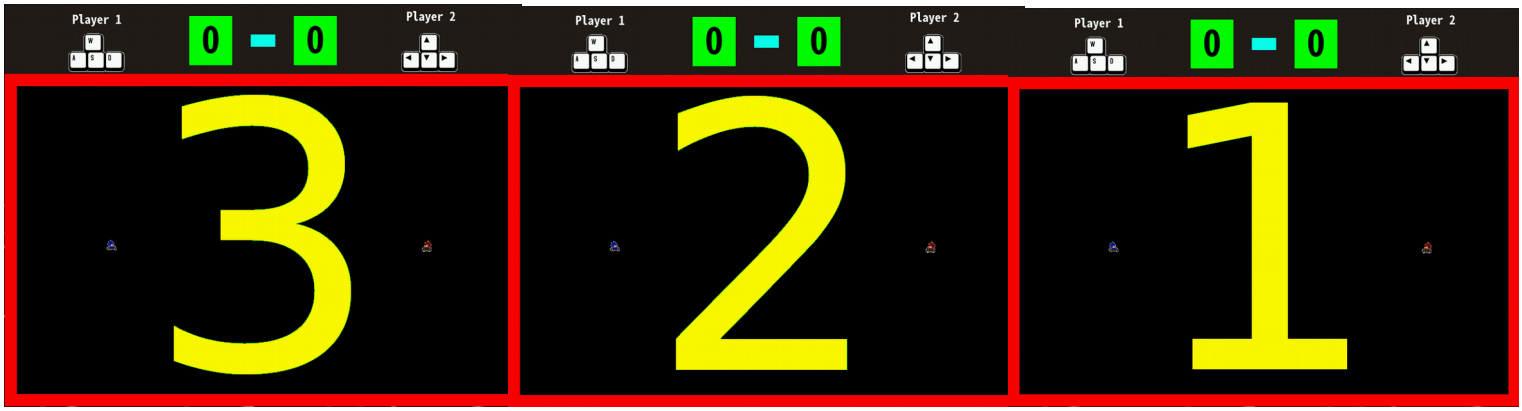
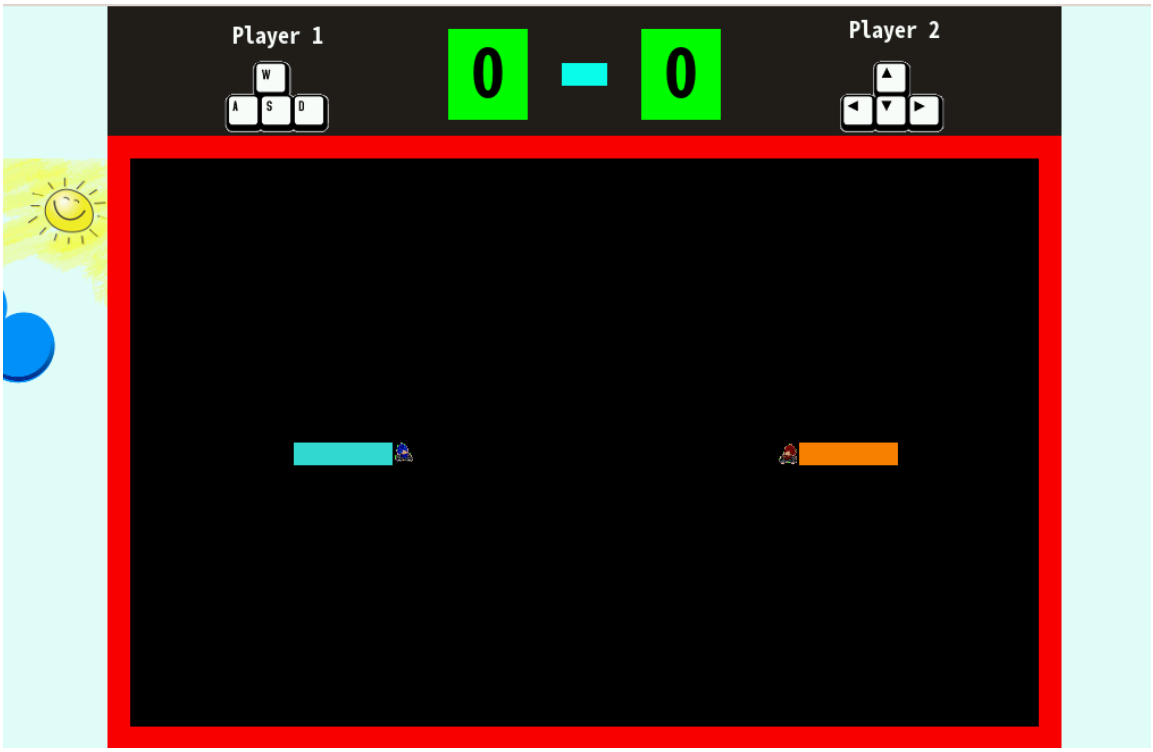


Fig. 5 –Aspeto gráfico da contagem decrescente antes do começo do jogo

Podem-se considerar três "secções" gráficas: o *scoreboard, o campo de jogo* e o fundo.

Fig. 6 – Aspeto Gráfico de “Local Multiplayer”



No decorrer do jogo, caso um dos jogadores perca a quinta ronda, é mostrado em texto “Player (número do jogador que ganhou) Wins”. Após o texto desparecer o programa retoma ao menu inicial.

Fig. 7 – Aspeto gráfico do ecrã
(caso o Player 1 ganhe o jogo)

Fig. 8 – Imagem caso o Player 2 ganhe o jogo
(caso o Player 2 ganhe o jogo)

- *Scoreboard:*

Esta secção encontra-se por cima do campo de jogo e contém informação sobre o resultado do jogo (que é atualizado a cada "ronda") e sobre as teclas que cada um dos jogadores devem utilizar para se poderem movimentar no jogo.

No lado esquerdo, estão indicadas quais as teclas que o Player 1 deve utilizar para se movimentar. No lado direito, o mesmo para o Player 2.

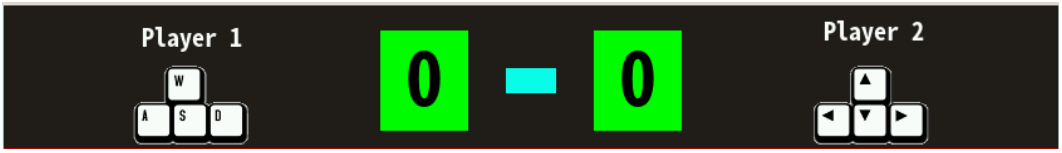


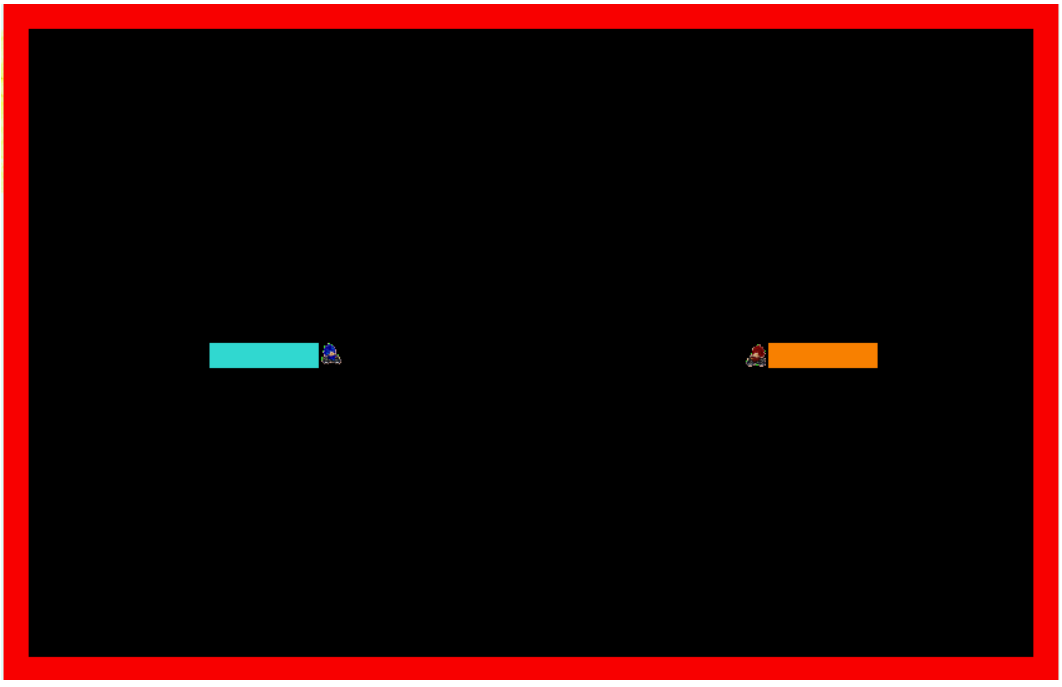
Fig. 7 – Aspeto Gráfico do *Scoreboard*

- Campo de Jogo:

O campo de jogo é delimitado por uma fronteira retangular vermelha onde toda a ação do jogo decorre. No seu interior, os jogadores movimentam-se deixando, a cada avanço, um rasto fixo.

No início do jogo, o jogador que se encontra no lado esquerdo (jogador azul) do campo corresponde ao Player 1 e o jogador que se encontra no lado direito (jogador laranja) corresponde ao Player 2.

Fig. 8 – Aspeto Gráfico do Campo de Jogo



Os jogadores controlam os seus bonecos pressionando as teclas-guia, como já foi atrás referido, no entanto, caso algum dos jogadores prima a tecla “ESC” o jogo acaba e o programa retoma ao menu inicial.

- Fundo:

Esta secção serve como “enchimento” do ecrã, visto que preenche toda a parte não preenchida pelo campo de jogo e pelo *scoreboard*. O fundo, altera também consoante a hora atual.

Fig. 9 - Fundo de Madrugada
(04h – 12h)

Fig. 10 – Fundo de Dia
(12h – 20h)

Fig. 10 - Fundo de Noite
(20h – 04h)

2.3. SINGLEPLAYER

No modo de jogo singleplayer, o aspeto gráfico é muito semelhante ao modo de jogo multiplayer.

Ambos apresentam o mesmo fundo, alterável consoante a hora do dia, e o mesmo campo de jogo. A diferença (gráfica) reside apenas no facto do *scoreboard* não possuir as teclas-guia para o Player 2, uma vez que este é guiado pelo próprio programa.

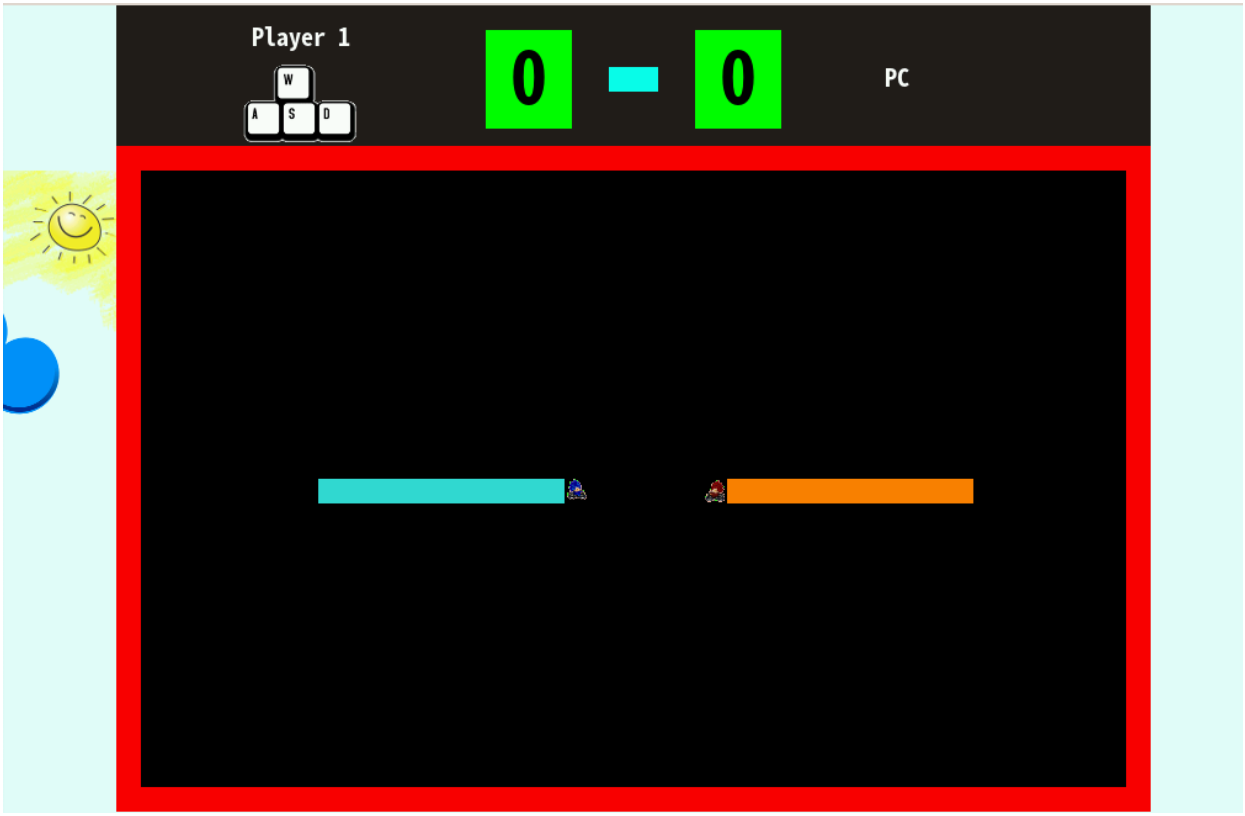


Fig. 10 - Aspeto gráfico do modo de jogo Singleplayer

Em mais detalhe, pode-se verificar a "secção" do *scoreboard* apresentando do lado esquerdo da pontuação as teclas-guia do Player 1 e do lado direito da pontuação a identificação do Player 2, neste caso o Programa, representado na figura abaixo.

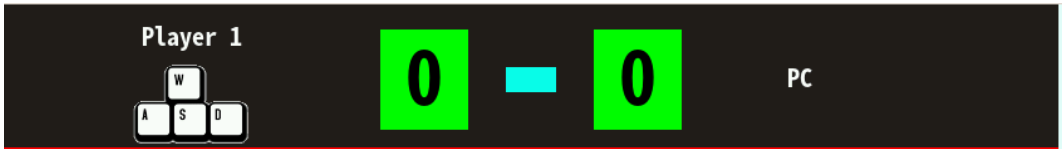


Fig. 11 – Aspeto Gráfico do *scoreboard* do modo de jogo Singlepayer

2.4. INSTRUCTIONS

Nesta opção do menu é apresentada uma imagem onde tem as instruções necessárias ao movimento dos jogadores, com as suas respetivas teclas-guia e com o significado de cada tecla.

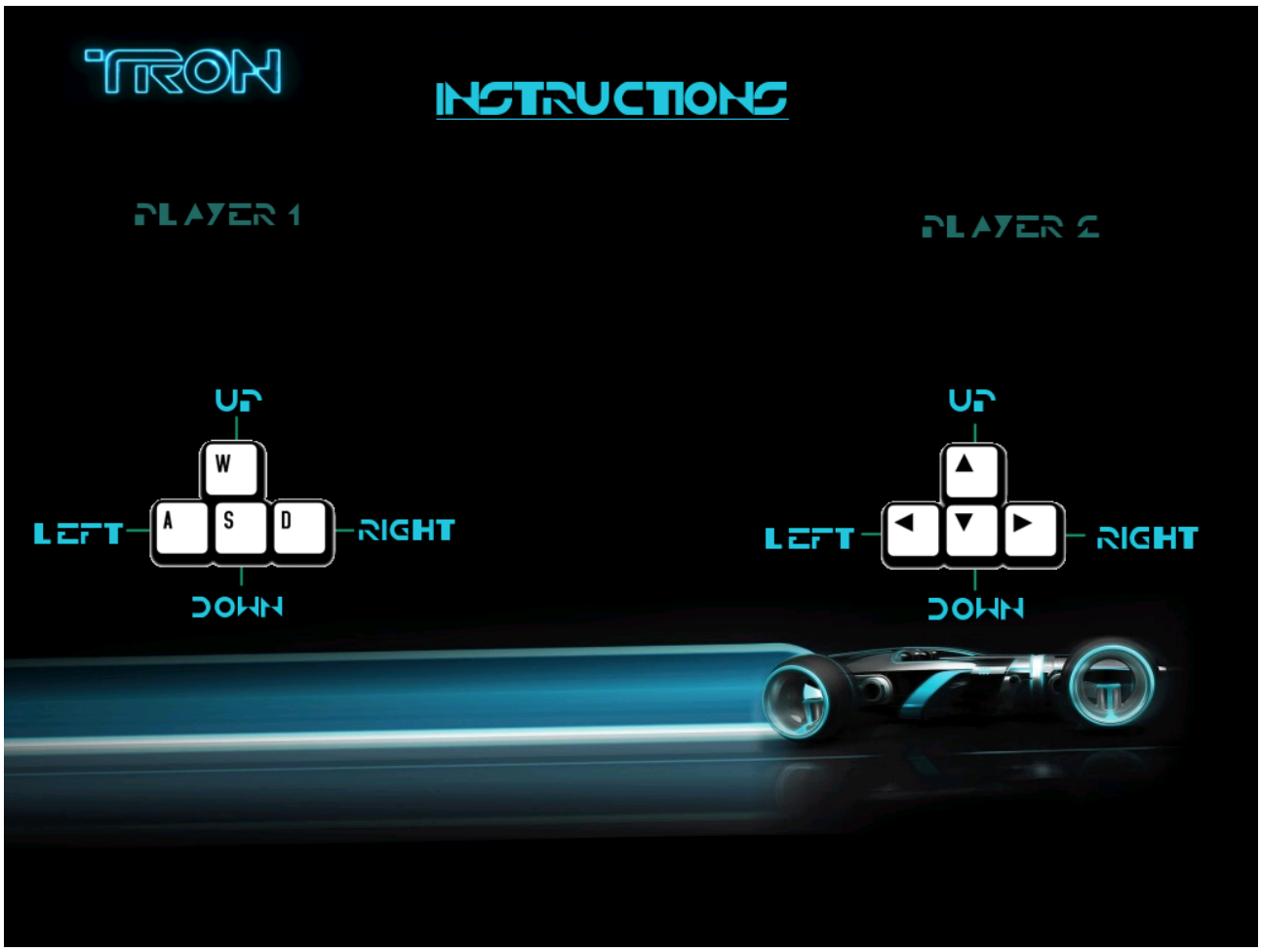


Fig. 12 – Imagem das intruções de jogo

2.5. EXIT

Caso o utiliazador pressione o botão “EXIT” o programa termina.

3. ESTADO DO PROJETO

Na construção do projeto implementámos os seguintes dispositivos:

Dispositivo	Função	Interrupções
KBD	Navegação no menu, saída dos modos de jogo e jogabilidade.	Sim
Rato	Navegação no menu.	Sim
Timer 0	Frame rate.	Sim
RTC	Obtenção da data e hora.	Não
Placa Gráfica	Apresentação gráfica do jogo.	Não

- **KBD:**

É usado para o movimento dos jogadores, para a navegação nos botões dos menus e, para sair dos modos de jogo e das instruções. Está definido no módulo “kbd” e é principalmente utilizado na função _keyboard_event do módulo “events” onde, dependendo do estado do programa, reconhece e executa os respetivos comandos associados às tecla permidas.

- **Rato:**

Este periférico é usado para a navegação no menu. Caso a posição do rato esteja por cima de um botão e o botão do rato premido seja o esquerdo, o programa muda de estado para o estado associado ao respetivo botão.

A definição das funções base e da estrutura (mouse_t) para este periférico estão definidas no módulo “mouse” e as funções onde este tem maior importância é mouse_event e tick, onde na primeira reconhece qualquer alteração que ocorra com o rato e a segunda, onde desenha o cursor do rato.

- **Timer 0:**

As interrupções do Timer 0 controlam o frame rate do jogo sem ser necessário alteração da frequência por defeito. É feito na função _main do ficheiro tron.c.

A definição das funções base encontram-se no módulo “timer”. Na função _tick do ficheiro events.c, o timer serve para a contagem decrescente de início de jogo e para o tempo de espera da imagem “PLAYER (jogador que ganhou) WINS!!!” de quando um dos jogadores ganha o jogo (multiplayer e singleplayer).

- **RTC:**

Este periférico é utilizado para obter a hora do dia de forma a poder atualizar o fundo de jogo. A definição das funções base encontram-se no módulo “rtc”, sendo que a função _level_draw do ficheiro level.c, lê a data e consoante a hora desenha a imagem corresponde ao dia/noite.

- **Placa Gráfica:**

A placa gráfica está encarregue de toda a parte visual do nosso projeto, sendo que o modo de vídeo por nós utilizado foi o 0x117, de dimensão 1024x768, 16 bits per pixel.

As funções relacionadas com a implementação da plaga gráfica estão nos módulos “vbe” e “video_gr”, sendo que para a sua implementação utilizámos a técnica de double buffering para uma implementação mais eficiente.

4. Organização do Código / Estrutura

A forma de organização que nós decidimos adotar para a realização do projeto foi que cada um realizava um módulo, e sempre que necessário entreajudavamos-nos.

Tron

Este módulo é o principal módulo do programa.

No ficheiro tron.h criou-se o **enum Game_State** que enumera os estados de jogo de acordo com a apresentação visual dos botões. Criou-se também a **struct Program** que guarda a informação base do programa, tais como as interrupções dos periféricos que estão a serem utilizados e o estado atual de jogo (**Game_State**).

No ficheiro tron.c estão definidas as funções base do programa, uma vez que todos os outros ficheiros do programa estão ligados a este módulo.

- Percentagem : 10%
- Membro Responsável : João Sá

Bitmap

Este módulo encarrega-se do carregamento, desenho e eliminação das imagens .bmp do nosso projeto.

Este módulo foi obtido a partir do site **https://difusal.blogspot.pt** do **Henrique Ferrolho**, com ligeiras alterações de forma a conseguirmos implementar o double buffer.

- Percentagem : 3%
- Membro Responsável : Francisco Machado Santos / João Sá

i8042

Este ficheiro apenas inclui a definição de constantes úteis para a escrita do código, tais como comandos específicos para o mouse e para o KBD.

- Percentagem : 1%
- Membro Responsável : Francisco Machado Santos

Events

Módulo que é utilizado para tratar qualquer evento que ocorra durante a execução do programa, tais como a seleção de um botão ou o uso das teclas para controlar o jogador ou sair do jogo. Este módulo é responsável pela interligação dos módulos uma vez que, dependendo do estado atual do jogo, chamadas funções dos diferentes módulos. É neste módulo que qualquer evento que ocorra é tratado.

- Percentagem : 17 %
- Membro Responsável : João Sá / Francisco Machado Santos

i8254

Este ficheiro inclui a definição de constantes, relacionadas com o Timer, úteis para a escrita do código.

- Percentagem : 1%
- Membro Responsável : João Sá

Kbd

Este módulo trata de todas as funções base relacionadas com o teclado.

No ficheiro kbd.h encontram-se definições de constantes de scancodes necessários para o controlo dos jogadores, para o abandono do jogo ou escolha dos botões.

No ficheiro kbd.c encontram-se as funções que lidam com as interrupções do teclado.

- Percentagem : 4 %
- Membro Responsável : João Sá / Francisco Machado Santos

Level

Este módulo encarrega-se de desenhar o fundo do jogo.

No ficheiro level.h foi criada a **struct level_t** que contém 4 bitmaps correspondentes ao fundo de jogo em quatro posições do ecrã, a área do fundo.

O ficheiro level.c encarrega-se do carregamento, da eliminação e do desenho dos bitmaps consoante a hora do dia.

- Percentagem : 5 %
- Membro Responsável : João Sá

Menu

Trata de todas as funcionalidades desponíveis no menu inicial.

No menu.h foram criadas duas **structs**, uma **button** que permite guardar informação de um botão, como a sua posição no ecrã e as suas imagens (normal/selecionado) e outra **menu** que contém toda a informação relacionada com o menu, como o número de botões e a imagem de fundo.

No ficheiro menu.c estão as funções responsáveis pela criação e desenho do menu inicial, bem como a sua eliminação.

- Percentagem : 10 %
- Membro Responsável : João Sá

Mouse

Este módulo é responsável por toda a informação relativa ao rato.

No ficheiro mouse.h encontram-se definidas algumas constantes úteis relacionadas com os pacotes do rato. Existe também uma **struct mouse_t** que contém informação relativa ao rato, útil para a sua implementação no menu.

- Percentagem: 6 %
- Membro Responsável : João Sá

Mp_local

Contém informação relativa ao jogo multiplayer.

É responsável pela atualização da posição dos dois jogadores e pelo desenho do rasto dos jogadores, bem como a criação e a eliminação do jogo multiplayer.

- Percentagem : 5 %
- Membro Responsável : João Sá

Players

Este módulo contém toda a informação relativa aos jogadores. Contém as funções responsáveis pela criação dos jogadores, pelo seu desenho (do boneco e do rasto), e pelo seu comportamento. O ficheiro players.c contém informação para o comportamento dos jogadores nos dois modos de jogo, sendo que no modo singleplayer gera caminhos aleatórios de forma a desviar-se dos obstáculos (Player 1, seu rasto e fronteiras) e no modo multiplayer as posições são atualizadas, verificando a cada atualização a posição onde se encontra e se é possível avançar ou mudar de sentido.

No ficheiro player.h criou-se a **struct player** que possui informação sobre a direção, a posição, velocidade, as imagens do jogador e ainda duas variáveis de control **wins** e **crash** que indicam respetivamente se o jogador ganhou, e se o jogador embateu contra algo.

- Percentagem : 17 %
- Membro Responsável : João Sá / Francisco Machado Santos

Rtc

Este módulo é responsável pela obtenção da data e hora do dia.

- Percentagem : 2 %
- Membro Responsável : Francisco Machado Santos

Scoreboard

Contém as funções que permitem cria, desenhar e eliminar a secção "*scoreboard*". No ficheiro scoreboard.h criou-se a **struct scoreboard_t** que contém os bitmaps dos números que vão ser mostrados no ecrã de jogo.

- Percentagem : 2 %
- Membro Responsável : João Sá

Singleplayer

Este módulo contém as funções que permitem criar, eliminar e desenhar o campo de jogo singleplayer.

- Percentagem : 5 %
- Membro Responsável : Francisco Machado Santos

Timer

Contém as funções que permitem controlar o Timer 0 (tratamento das interrupções) bem como o desenho do contador no início de jogo. Para tal, criou-se a **struct timer_struct** que contém as imagens dos segundos (3,2,1) e duas variáveis de controlo **contador_start** e **contador_end** de forma a mostrar o a imagem do segundo, segundo-a-segundo.

- Percentagem : 4 %
- Membro Responsável : João Sá / Francisco Machado Samtos

Vars

Este módulo serve apenas para definir constantes auxiliares. Criou-se o **enum Direction** de forma a enumerar os sentidos dos movimentos dos jogadores.

- Percentagem : 1 %
- Membro Responsável : Francisco Machado Santos / João Sá

Vbe

Este módulo contém funções que retornam a informação sobre as capacidades do modo de vídeo.

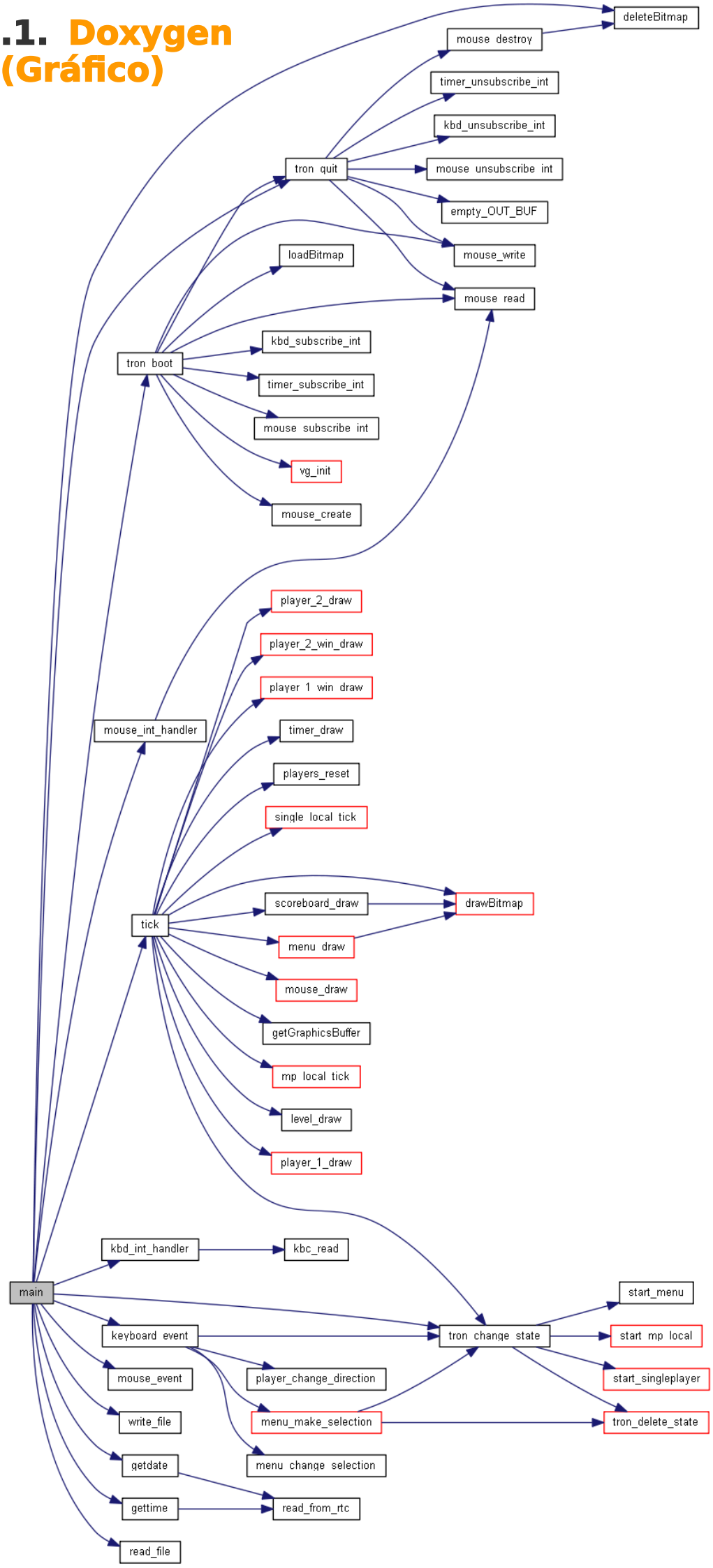
- Percentagem : 4 %
- Membro Responsável : João Sá / Francisco Machado Santos

Video_gr

Este módulo contém funções que retornam a informação do modo gráfico (dimensão), bem como o buffer. Contém ainda as funções que permitem iniciar o modo de vídeo, sair e desenhar o rasto dos jogadores (_draw_pixel).

- Percentagem : 5 %
- Membro Responsável : João Sá / Francisco Machado Santos

4.1. Doxygen (Gráfico)



5. Detalhes da Implementação

Na construção do projeto, cerca de 90% dos periféricos que nós utilizámos foram lecionados nas aulas de LCOM. Existiram algumas dificuldades na implementação de algumas funcionalidades dos periféricos, contudo após uma pesquisa mais aprofundada conseguimos contornar esses obstáculos e ultrapassá-los.

Implementação:

➔ Placa Gráfica :

Melhorámos, significativamente, a implementação da placa gráfica no projeto em relação ao lab5. Decidimos implementar a técnica "*double buffering*" , em vez da implementada no lab5 ("*linear frame model*"), uma vez que é bastante mais eficiente no desenho de imagens, sendo que desta forma reduz o risco de ocorrer

➔ KBD :

A implementação do teclado, foi a mesma que a implemenada no lab3, sendo que o teclado tem um papel fulcral no nosso jogo, uma vez que a escolha do sentido do movimento dos jogadores só é possível recorrendo ao teclado. A cada interrupção do teclado, o programa (dependo do estado em que se encontra), executa o comando correspondente.

➔ Timer :

Optámos por implementar o timer da mesma forma que no lab2 pois apenas usamos as interrupções do timer 0. O timer 0 possui um papel significativo na implementação do programa é usado para o "refresh rate", bem como a **struct timer_struct** que serve de contador uma vez que antes de começar qualquer modo de jogo, espera-se três segundos antes de os jogadores se começarem a movimentar.

➔ RTC :

O RTC, é nos útil para verificar a hora do dia. Consoante a hora, alteramos o fundo de jogo.

➔ Rato :

O rato foi implementado tendo como base o lab4. Foi criada uma **struct** designada por **mouse_t** que contém informação relativa à posição do rato, à sua apresentação gráfica e se alguns dos butões foi pressionado. É através de uma variável deste tipo que é possível desenhar e verificar a posição do rato, crucial para a implementação do menu inicial.

Um grande desafio com o qual nos deparámos foi, no modo **Singleplayer**, o de movimentar o Player 2 no jogo, uma vez que o seu movimento tem de parecer natural. Foi então, no ficheiro **players.c**, onde criámos uma função **_player_2AI** que se encarrega de verificar se o Player 2 se encontra perto de um rasto do, ou do próprio, Player 1. Caso afirmativo, o Player 2 verifica para onde pode se deslocar. Se se puder deslocar livremente, então é gerado um número aleatório que irá definir o novo sentido do jogador.

No final do jogo, após clicar no botão “EXIT”, o programa sai do modo de vídeo e imprime a data e a hora de início do jogo. Esta informação está guardada num ficheiro de texto denominado por “example”. A escrita e leitura do ficheiro está implementada no ficheiro tron.c com os nomes _write_file e _read_file.

6. Avaliação

A nosso ver, a unidade curricular encontra-se bem estruturada, dando-nos a conhecer e a entrar em contacto com a programação de "baixo nível" (C / Assembly) permitindo-nos aprofundar o nosso conhecimento sobre como o computador lida com os difirentes periféricos.

Achamos também que, embora nas aulas teóricas seja explicado o código, houvesse um melhor esclarecimento e melhor orientação de como começar a escrever o código, pois por várias vezes não conseguíamos perceber por onde começar.

Por último, achamos que para as pessoas que têm aula segunda-feira de tarde, é muito injusto, uma vez que várias vezes tínhamos de acabar o lab com matéria que íamos dar umas horas antes.

7. Notas / Créditos

Gostaríamos de agradecer ao Henrique Ferrolho pelo tutorial existente no seu blog, pois ajudou-nos a perceber como usar o módulo do bitmap.

Queremos ainda agradecer aos elementos do grupo 25 da turma 6, Bernardo Barbosa e Duarte Carvalho, por nos terem ajudado com a função do greenscreen (_drawBitmap_greensccreen, declarada no módulo bitmap).