Faculdade de Engenharia da Universidade do Porto

MIEIC

Laboratório de Computadores

Tron Arcade

Relatório Final

Grupo T4G12:

João Seixas up201505648@fe.up.pt

Bernardo Ramos up201505092@fe.up.pt

2016/2017

Índice

1. Instruções de Utilização

1.1 Menu

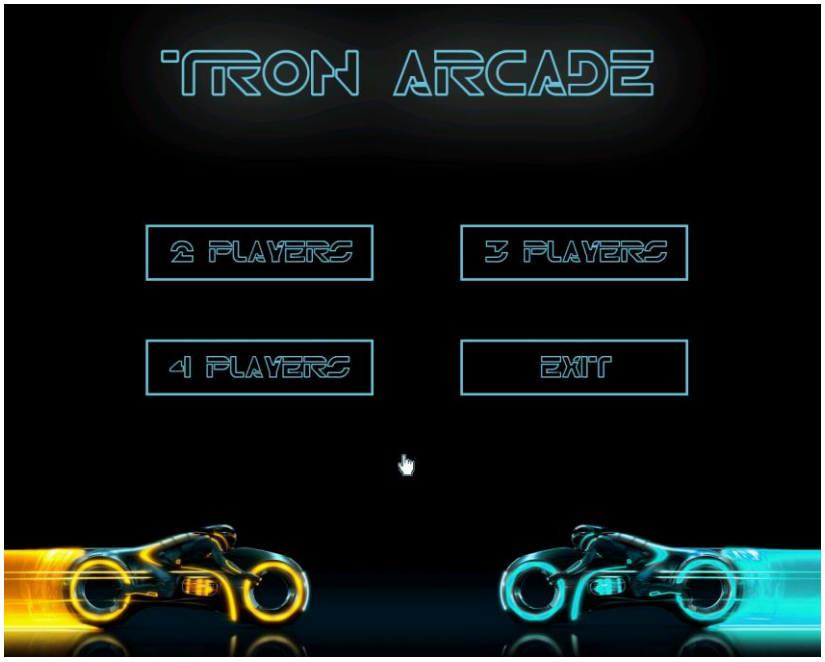

Figura 1 - Menu

O programa inicia com um menu no qual tem 4 opções: Jogo com 2, com 3 ou com 4 *players* e um botão para fechar o programa (*Exit*). A escolha das opções é feita através do movimento e botões do rato, mas também é possível selecionar com o teclado (tecla Esc para Sair, tecla 2 para Jogo com duas pessoas, tecla 3 para três pessoas e a tecla 4 para quatro pessoas).

1.2 Jogo

Entrando numa das opções de jogo é apresentado uma imagem (Figura 2) com as respetivas teclas/botões (uma vez que o quarto jogador irá orientar com os botões do rato) para cada jogador orientar o seu rasto, assim como outras opções como pausar o jogo ou retornar ao menu. Para começar é preciso premir a tecla espaço.

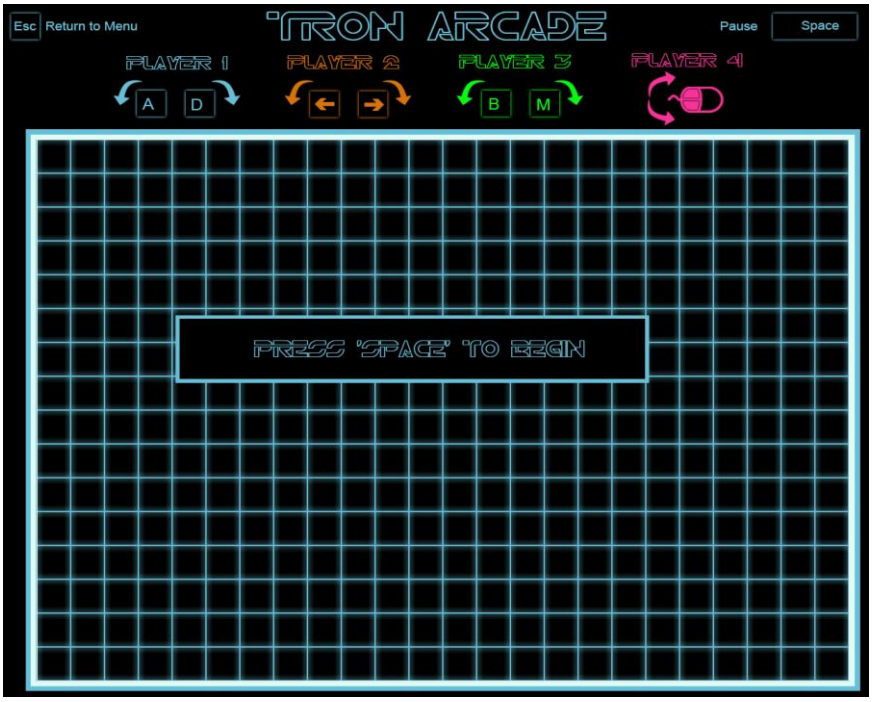

Figura 2 - Começar o Jogo

Cada jogador deixa o seu rasto (diferentes cores para diferentes jogadores) e perde se colidir com qualquer rasto (seu ou de outro jogador) ou contra as bordas da área de jogo. Ganha quem não colidir e que todos os outros jogadores tenham colidido. Na figura abaixo (Figura 3) podemos ver que o jogador 3 colidiu com o 4 e os jogadores 1 e 2 ainda estão em jogo. Para orientar o rasto, o botão esquerdo do rato e as teclas A, Seta Esquerda e B servem para fazer uma rotação no sentido anti-horário e o botão direito do rato e as teclas D, Seta Direita e M servem para fazer uma rotação no sentido horário.

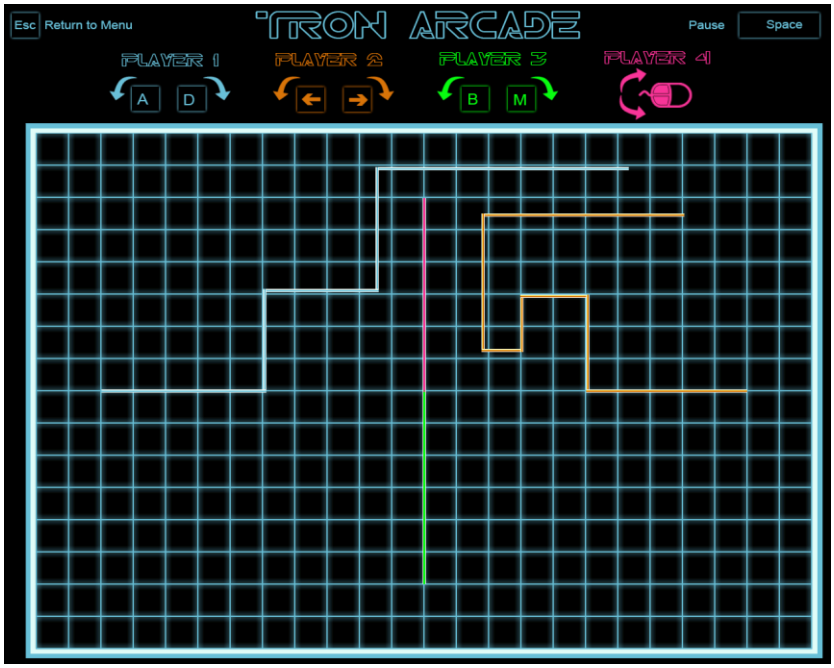

Figura 3 - Jogo com 4 jogadores

Quando alguém ganha aparece um menu a dizer quem ganhou e para premir o espaço se quiserem voltar a jogar.

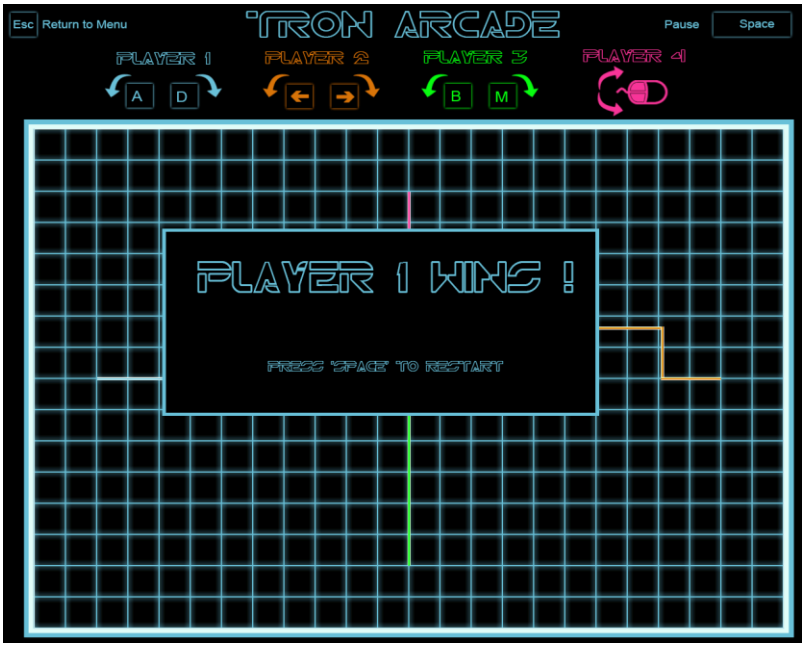

Figura 4 - Vencedor

2. Estado do Projeto

No projeto não foi implementado o RTC nem a Porta de Série. O projeto está funcional. Em relação à especificação, foi bastante alterado, uma vez que não introduzi o modo *singleplayer* nem adicionei obstáculos.

2.1 Dispositivos de Entrada/Saída

Dispositivo	Utilidade	Modo de Utilização
Timer	Controlo dos *frame rates*	Interrupções
KBD	Movimento dos jogadores e saída do programa	Interrupções
Mouse	Seleção nos menus e movimento de um jogador	Interrupções
Video Graphics	Representação gráfica do jogo	Polling com Interrupções do Timer
RTC	Não utilizado	
UART	Não utilizado	

2.1.1 Timer

Usado para controlo dos frame rates, ou seja, para imprimir o rasto de cada players no jogo ou o cursor do rato no menu. É usado nas funções playgame e timer_intrhandler. O código de subscrições está no ficheiro otherlabs.c.

2.1.2 KBD

Usado para controlar o movimento dos jogadores, para saída do programa, pôr o jogo em pausa e em paralelo com o rato escolher o modo de jogo. É usado como clicar e não ficar a premir, uma vez que um jogador tem de manter sempre o movimento enquanto não perde. É principalmente usado nas funções change_player_state e state_handler. O código de subscrições e de recolher a informação do registo do KBC está implementado no ficheiro otherlabs.c.

2.1.3 Mouse

Usado com *state machines* para seleção de opções e movimento do cursor no menu e é também usado os botões para controlar um jogador no jogo de 4 jogadores. É principalmente usado nas funções mouse_mov_handler e mouse_st_handler. O código relativo a subscrições e de enviar os modos para o registo esta implementado no ficheiro otherlabs.c.

2.1.4 Video Graphics

É usado o *video mode* 0x11A com resolução 1280x1024 com 64K de cores com RGB 5:6:5. Não é usado *double buffering* apesar de estar implementado, uma vez que da forma que foi desenvolvido, o programa apenas vai sendo desenhado a posição atual do jogador (o rasto fica sempre), não havendo necessidade de voltar a redesenhar o fundo e por isso mesmo seria uma utilização que afetaria a fluidez e consumiria muito mais recursos no seu processamento. A deteção de colisões entre os rastos dos jogadores e as bordas da área do jogo é feita através da comparação de cores (todas as bordas são brancas). Não é a melhor solução, mas é a mais rápida. As funções relacionadas estão no ficheiro video_gr.c e vbe.c.

3. Organização e Estrutura do Código

3.1 game.c

Este módulo é o responsável pela lógica e funcionamento do jogo. Contém o *loop* das interrupções assim como os *handlers* todos e grande parte das funções chamadas por estes.

Feito por: João Seixas – 100%

Peso no projeto: 35%

3.2 otherlabs.c

Este módulo contém todas subscrições e outras funções relacionadas com o teclado, rato e timer desenvolvidas nos labs das aulas práticas durante o semestre.

Feito por: João Seixas – 100%

Peso no projeto: 10%

3.3 read_bitmap.c

Este módulo contém todas as funções e estruturas para poder ler e manipular bitmaps. O código utilizado foi feito pelo Henrique Ferrolho e foi obtido a partir de http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html .

Peso no projeto: 10%

3.4 tools.c

Este módulo contém no *header file* tools.h todas as estruturas do jogo (*structs* e s*tate machines*) e constantes necessárias ao funcionamento de todo o projeto. Está implementado uma *struct* para todo o programa – game_t – que contém todos os jogadores, o rato, os bitmaps e ainda variáveis e *flags* necessárias. O módulo ainda contém funções para inicializar os jogadores

(cada jogador é um objeto de uma *struct*) e o rato (objeto de uma *struct* com *state machines*) entre outras. Ainda contém a função rgb que foi obtida a partir de http://stackoverflow.com/questions/27194568/rgb-color-converting-into-565-format pelo utilizador PiotrK.

Feito por: João Seixas – 100%
Peso no projeto: 20%

3.5 Tron.c

Este módulo só contém a função main.
Feito por: João Seixas – 100%
Peso no projeto: 5%

3.6 vbe.c

Desenvolvido no lab5.
Feito por: João Seixas – 100%
Peso no projeto: 5%

3.7 video_gr.c

Este módulo contém funções desenvolvidas no lab5 e funções para pintar o rasto dos jogadores e as bordas do jogo.

Feito por: João Seixas – 100%
Peso no projeto: 15%

3.8 Diagrama de Chamada de Funções

O gráfico de Chamada de Funções (ver Figura 5) foi gerado pelo Doxygen e pelo GraphViz.

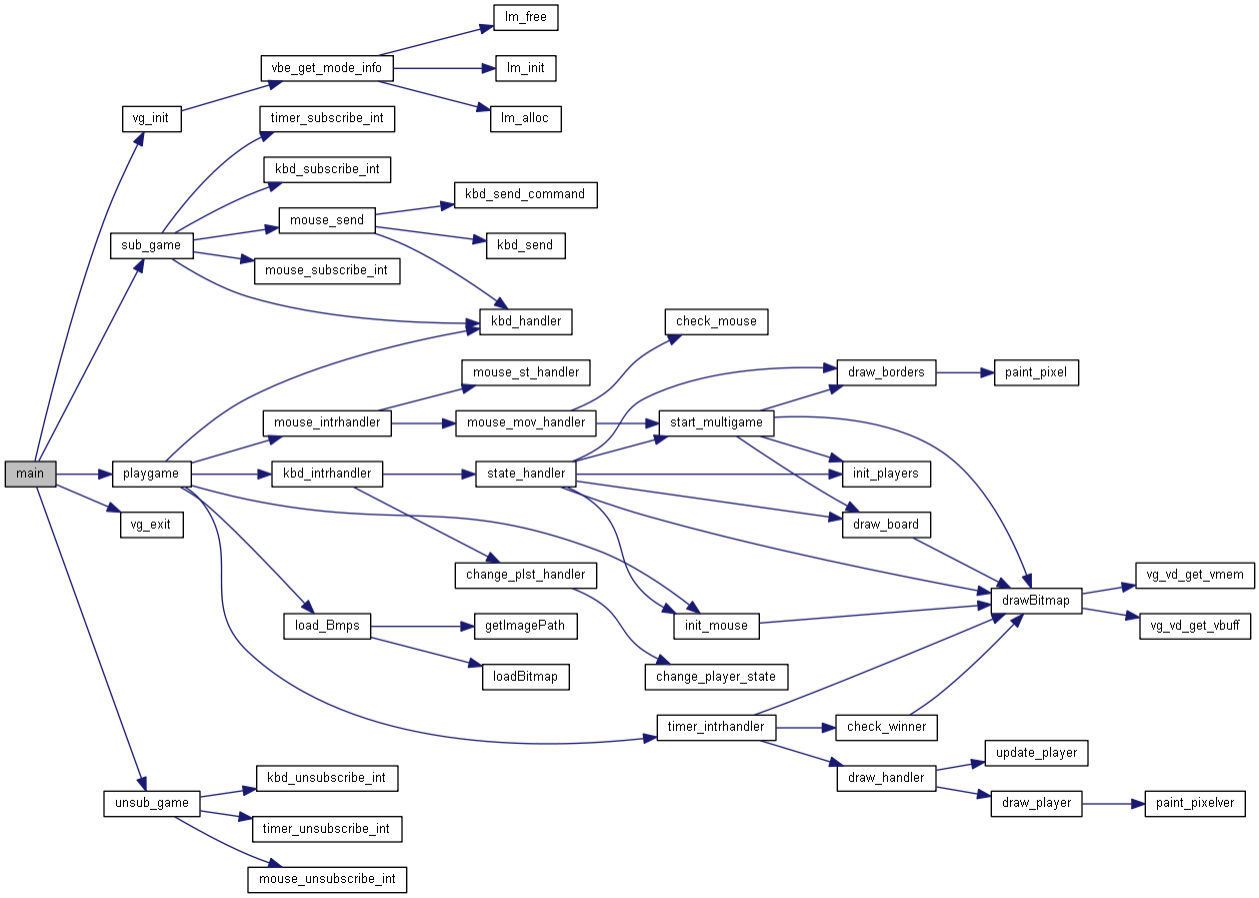

Figura 5 - Diagrama de Chamda de Funções

4. Detalhes da Implementação

No projeto foi implementada uma *struct* para conter toda a informação necessária ao funcionamento de todo o programa. Na função main é criado um objeto desta struct que vai sendo passado como argumento nas funções, de forma a poder ir atualizando a informação.

Esta *struct* contém todos os bitmaps, quer para o menu, quer para o fundo do jogo, os quais são guardados logo na inicialização do programa. Contém também *struct*s para os jogadores e para o rato.

A *struct* dos jogadores contem informação sobre a sua posição atual, sobre os *scancodes* das teclas responsáveis pelo seu movimento, um bitmap correspondente ao menu que aparece para exibir quem ganhou e ainda uma máquina de estados para saber em que direção o jogador está a ir. No inicio do desenvolvimento do trabalho era usado 4 teclas para o movimento do jogador, mas uma vez que um jogador tem de manter o movimento (não pode simplesmente parar) foi optado o uso de apenas 2 teclas para mudar o movimento de um jogador (no sentido horário ou anti-horário).

A struct do rato contém a sua posição (a do canto superior esquerdo do bitmap), máquinas de estados para o botão esquerdo e direito para se poder saber se estão a ser pressionados ou não e ainda uma variável para saber se houve alterações no rato (para saber se é necessário redesenhar).

5. Conclusão

Apesar de bastantes diferenças entre a especificação e o projeto final, grande parte dos objetivos deste trabalho foi implementado com sucesso. Faltou apenas implementar o RTC e a Porta Série.

5.1 Avaliação

Apesar de o trabalho ser do grupo T4G12, apenas o aluno João Seixas (up201505648) trabalhou neste projeto. O outro elemento não obteve frequência por faltas dadas e não participou no projeto.

6. Instruções de Instalação

Para instalação e funcionamento correto do projeto (uma fez que são usados *paths* para ir buscar as imagens usadas no jogo) basta mover o ficheiro dentro da pasta conf para /etc/system.conf.d, estar no diretório /home/lcom/lcom1617-t4g12/proj/src/ e executar o comando "make", e de seguida o comando "service run `pwd`/Tron". Note que apenas para o comando "make" não precisa de permissões *root*.