U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Laboratório de Computadores

TURMA 3 – GRUPO 8

Realizado por:

Pedro Miguel Sampaio Ferreira Machado up201906712

Luís Filipe Carvalhais dos Santos de Matos up201905962

Sofia Ariana Moutinho Coimbra Germer up201907461

Índice

Introdução

A nossa escolha para desenvolver um projeto final recaiu na criação de um jogo 2D numa perspetiva de cima para baixo. O jogo tem dois modos, o modo *SinglePlayer* e o modo *MultiPlayer*. Denotar que o jogo Whac-A-Mole é um popular jogo de *arcade* criado no Japão no ano de 1975.

A essência do jogo consiste em acertar em toupeiras sempre que estas estiverem fora do seu buraco, tendo em conta que a rapidez com que estas entram e saem gera uma dificuldade em lhes acertar. Importante referir que escolhemos 6 como o número de buracos dos quais podem sair ou entrar toupeiras.

No modo *SinglePlayer*, tal como o nome indica, existe apenas um jogador e, assim sendo, todas as toupeiras que forem acertadas contarão para a pontuação do mesmo.

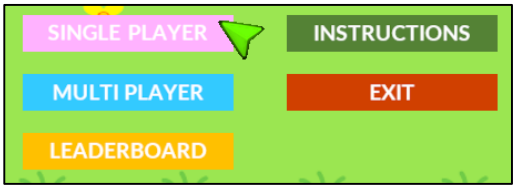
Já no que diz respeito ao modo *MultiPlayer*, existem dois jogadores que estão a jogar de forma competitiva entre eles. As toupeiras estão sincronizadas, pelo que o primeiro jogador a acertar na toupeira será o que obterá essa pontuação. No final do jogo, ganha o jogador que obtiver melhor precisão, ou seja, uma maior pontuação.

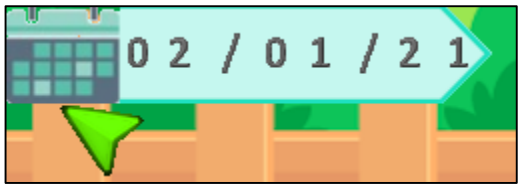
1. Instruções de Utilização

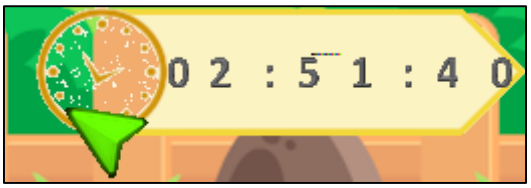
1.1 Menu Inicial

Assim que o jogo se inicia, é mostrado no ecrã um pequeno menu constituído por 5 botões. Através deles o jogador tem a possibilidade de: escolher o modo de jogo a ser selecionado (*SinglePlayer* ou *MultiPlayer*); consultar a *leaderboard* (escolhendo o botão *Leaderboard*) que contem os 5 melhores scores obtidos; consultar as instruções do jogo (escolhendo o botão *Instructions*) ou apenas sair do jogo (escolhendo o botão *Exit*).

Para navegar neste pequeno menu devem ser usados os movimentos do rato para deslocar o cursor, sendo que sempre que o cursor estiver sobre um botão este ficará mais claro (sinalizando esta informação), e os cliques no botão esquerdo do rato que têm a função de seleção de um dos botões.

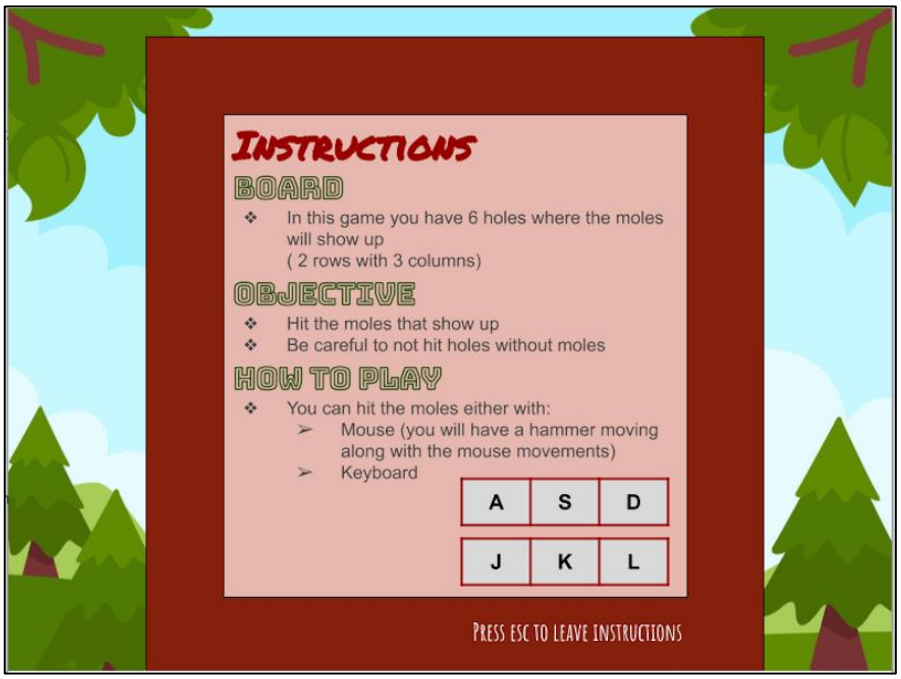






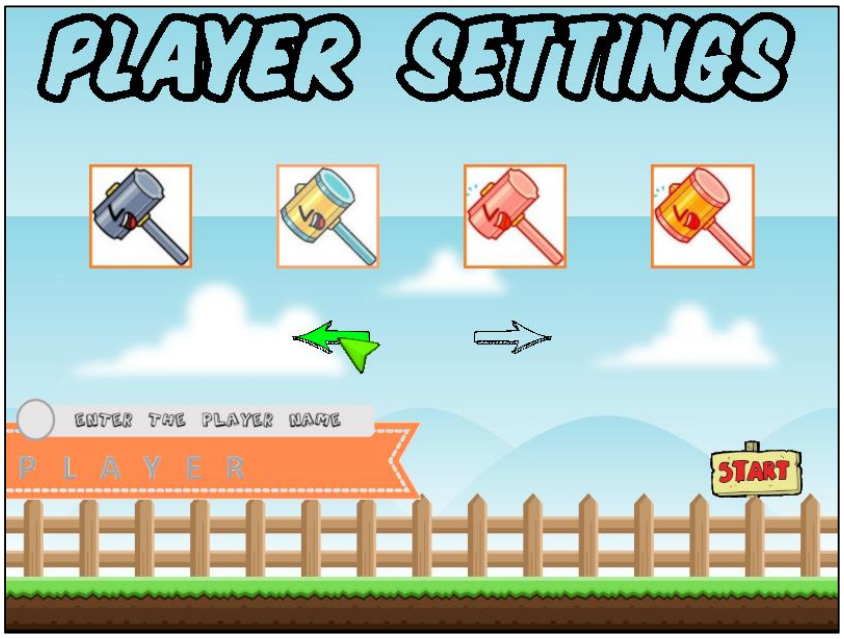
O jogador tem ainda a possibilidade de consultar a data atual deslocando o cursor para cima do calendário que aparece no topo esquerdo do ecrã (que passará a mostrar o pretendido) e por fim, de forma análoga, poderá consultar a hora atual utilizando o relógio que aparece também no topo do ecrã.

1.2 Instruções



Quando é selecionado o botão *Instructions* o jogador é encaminhado para este ecrã. Aqui são mostradas umas pequenas instruções de como funciona o jogo. Para voltar ao menu inicial o jogador apenas precisa de clicar na tecla *esc*.

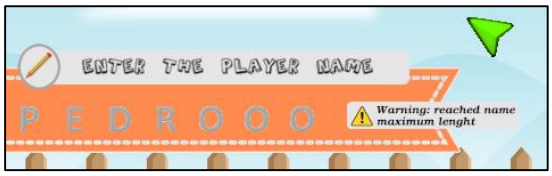
1.3 Escolha do Player avatar e nome



Neste ambiente o jogador tem a oportunidade de **escolher qual a imagem do martelo** que pretende utilizar no jogo, tendo 4 opções à sua escolha, e ainda **definir qual o seu nome**, para posterior registo na *leaderboard* caso o seu *score* seja suficientemente bom para tal.

Quanto à escolha da imagem, existem 4 opções à disposição, sendo que por defeito a imagem que aparece selecionada é a primeira. Para alterar a imagem selecionada o jogador dispõe de duas setas. Como podemos ver, caso o cursor esteja por cima de uma das setas esta ficará a verde mostrando assim essa informação. Para alterar a imagem selecionada para a da esquerda o jogador simplesmente precisa de mover o cursor até à seta da esquerda, que ficará verde, e usar o clique do botão esquerdo para confirmar. De forma idêntica terá de proceder para alterar a imagem selecionada para a da direita.

De salientar que caso a imagem selecionada seja a mais à esquerda/direita o clique do botão do esquerdo quando o cursor se encontra sobre a seta da esquerda/direita não fará mudança nenhuma.



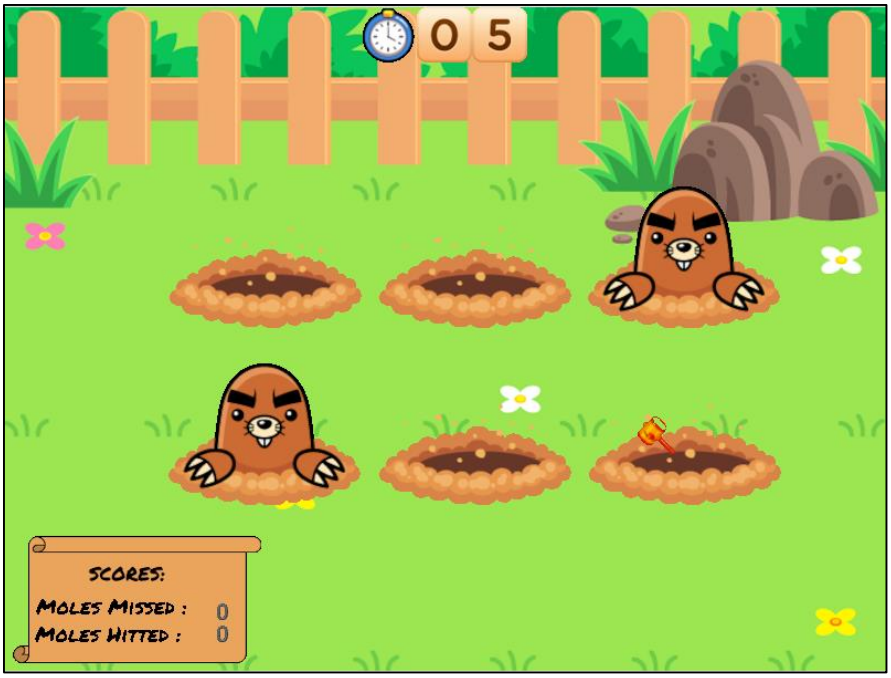
Ao nível da escolha do nome do jogador, é possível escolher um nome de comprimento máximo igual a 7 letras, sendo que por defeito o nome que está escrito é 'PLAYER'. Para alterar o nome, o jogador terá de utilizar não só o teclado, mas também o rato. Inicialmente terá de deslocar o cursor do rato até que este se encontre sobre a caixa onde está escrito o nome. Para sinalizar o sucesso desta ação, irá aparecer um pequeno lápis como forma de dar a informação ao jogador que está pronto a editar o seu nome. Neste momento, usando o teclado, o jogador é capaz de apagar (usando a tecla *backspace*) um carater e escrever quantos desejar, respeitando o comprimento máximo permitido. De forma a ajudar o utilizador, caso o comprimento máximo do nome seja atingido irá ser mostrado um pequeno aviso, que só desaparecerá caso o jogador apague uma letra ou carregue na tecla *enter* dando a informação que terminou a edição do seu nome.

Após estas duas personalizações o jogador está pronto a jogar e para tal apenas necessita de descolar o cursor do rato até este se encontrar sobre o botão *start*, que ficará um pouco mais claro quando tal acontecer, e de seguida clicar no botão esquerdo do rato.

1.4 Waiting

Assim que o jogador seleciona o botão *MultiPlayer* e depois de escolher as suas *settings* em *player settings* será encaminhado para este ecrã. Aqui apenas é dito que nesse momento o jogador terá de esperar até que o outro jogador se conecte. Para sair do jogo, caso assim o deseje, o jogador apenas precisa de clicar na tecla *esc*.

1.5 SinglePlayer



Quando se inicia o jogo escolhendo o modo *SinglePlayer* é mostrado no topo do ecrã, junto a um pequeno ícone de um relógio, o tempo de jogo decorrido. São apresentados também 6 buracos, organizados em 2 linhas de 3 buracos cada. No canto inferior esquerdo é mostrada a tabela de pontuação.

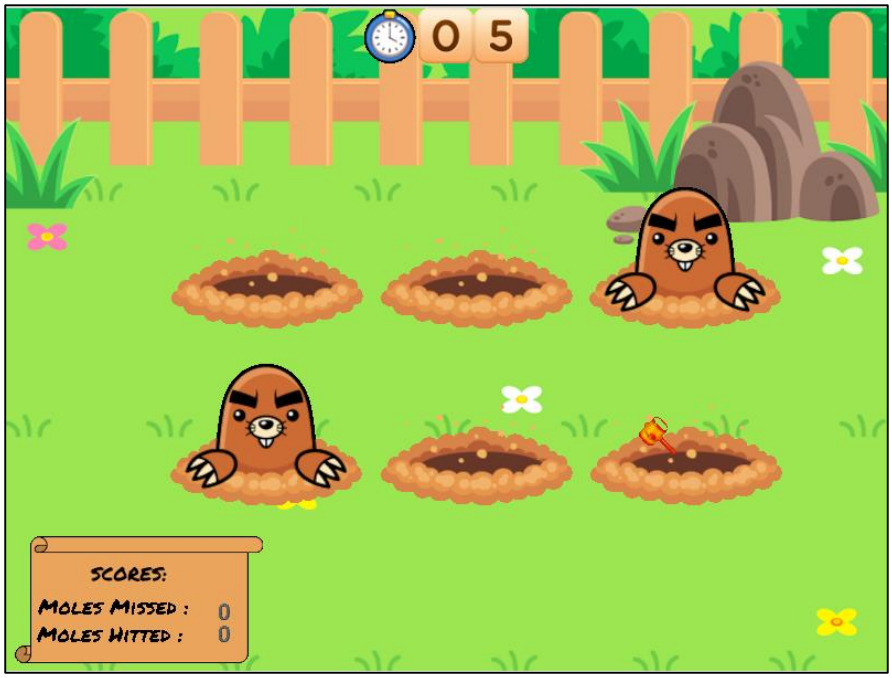
O jogo consiste em acertar nas toupeiras que vão aparecendo ao longo do tempo e para tal o jogador pode recorrer quer o rato, quer o teclado. Quanto ao rato, para acertar numa toupeira o jogador necessita apenas de deslocar o martelo até que este se encontre sobre a toupeira que pretende acertar, e clicar no botão esquerdo do rato. Quanto ao teclado, a sua utilização baseia-se na atribulação de uma tecla de ação a cada buraco (informação disponível nas instruções do jogo) tendo em conta que para acertar numa toupeira o jogador apenas precisa de premir a teclada a ela associada.

A cada clique no botão esquerdo do rato ou numas das teclas de ação, será verificado se o mesmo acertou na toupeira ou não. Em caso de sucesso o número de toupeiras acertadas que irá aumentar, caso contrário será o numero de toupeiras falhadas a aumentar.

Estes dois números servem de base ao cálculo da pontuação do jogador, que se baseia na sua precisão calculada da seguinte maneira: toupeiras acertadas/(acertadas+falhadas). Quantas mais toupeiras falhadas menor será a sua precisão, logo menor será também a sua pontuação. Quanto mais toupeiras acertadas maior será a sua precisão, logo maior será também a sua pontuação.

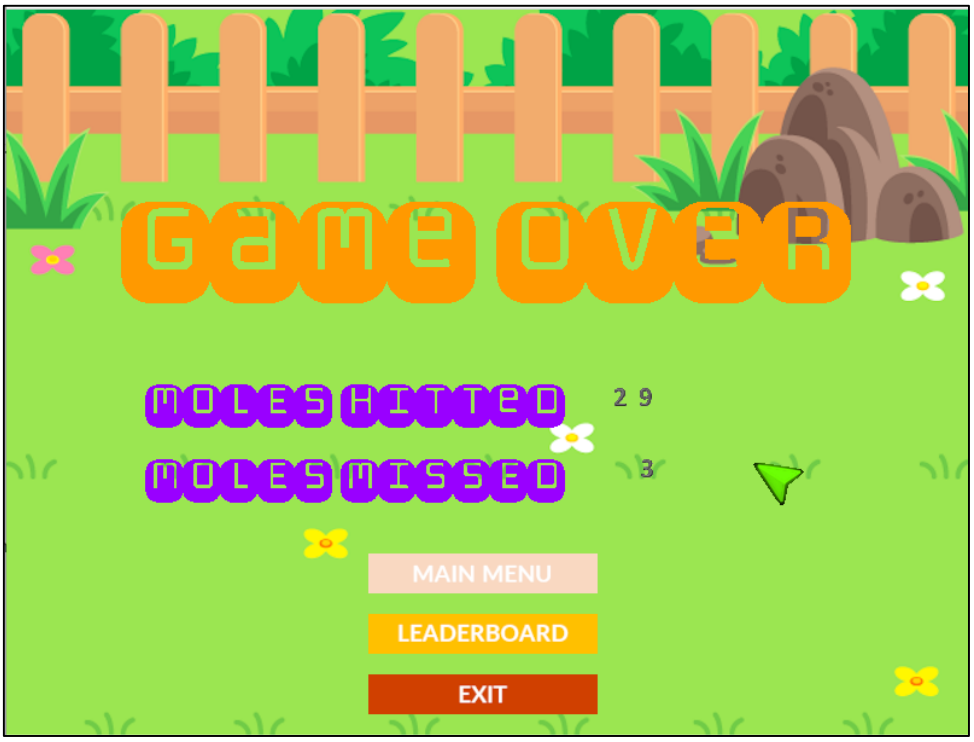
No final do jogo, quanto o tempo de jogo atingir o valor definido o jogador será encaminhado para o ambiente de *GameOver*.

1.6 MultiPlayer



Caso o modo selecionado tenha sido *multiplayer* o jogador irá competir com o seu adversário. As toupeiras irão subir e descer de forma sincronizada no ecrã de ambos os jogadores. Assim que um jogador acerte numa toupeira, utilizando para isso o teclado ou o rato, a toupeira irá descer para ambos os jogadores, mas a pontuação será atribuída apenas ao jogador que lhe acertou. A forma geral de jogar é exatamente igual ao modo *singleplayer*.

1.7 Game over



Assim que o jogo termina o jogador é encaminhado para este ecrã. Aqui é-lhe mostrado o seu desempenho no jogo que realizou, número de toupeiras que acertou e que falhou.

Caso o seu score seja suficiente para substituir um dos 5 registos presentes na *leaderboard*, o jogador será adicionado à mesma e aparecerá um balão a subir no ecrã, sinalizando essa conquista.

Existe ainda um pequeno menu com três opções: regressar ao menu principal (carregando no botão *Main Menu*), ver a lista das 5 melhores pontuações (carregando no botão *Leaderboard*) ou então sair do jogo (carregando no botão *Exit*).

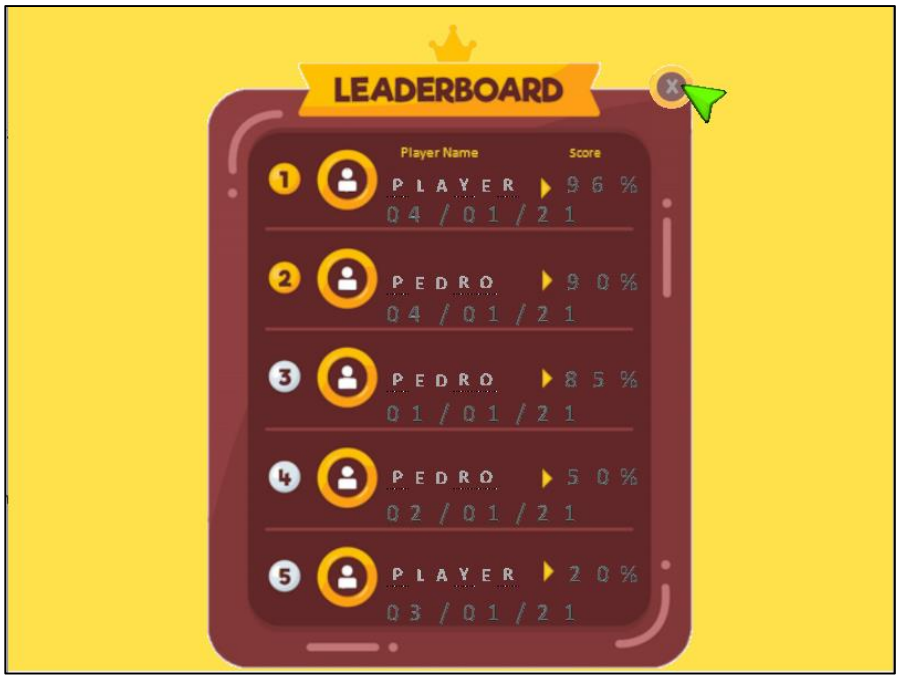
1.8 Game win

Caso o modo escolhido tenha sido *multiplayer* e o jogador tenha sido o vencedor, será então encaminhado para este ecrã em que essa informação lhe é dita. Para sair do jogo, caso assim o deseje, o jogador apenas precisa de clicar na tecla *esc*.

1.9 Game Lost

Caso o modo escolhido tenha sido *multiplayer* e o jogador tenha sido o perdedor, será então encaminhado para este ecrã em que essa informação lhe é dita. Para sair do jogo, caso assim o deseje, o jogador apenas precisa de clicar na tecla *esc*.

1.10 Leaderboard



Quando é carregado no botão *Leaderboard* o jogador é encaminhado para este ecrã. Aqui são mostrados os registos dos 5 melhores resultados obtidos pelos jogadores anteriores. Cada registo é caracterizado pelo nome do jogador que realizou aquela pontuação, a pontuação obtida e ainda a data em que essa pontuação foi obtida.

Para sair deste ecrã o jogador apenas precisa de mover o cursor até ao pequeno botão redondo com um 'x' que se encontra no canto superior direito da tabela e clicar no botão direito do rato.

1.11 Exit

Quando o utilizador decide sair do jogo encontra uma mensagem de despedida (*Good-Bye*) e uma toupeira animada que se desloca no fundo do ecrã da esquerda para a direita. Esta animação dura 3 segundos sendo que no final, o jogo termina. Para além disso, são exibidos ainda os créditos do jogo junto ao canto superior direito.

2. Project Status

Dispositivo	Funcionalidade	Interrupções
Timer	Controlar a frame-rate	Sim
Keyboard	Acertar em moles e editar o player name	Sim
Graphics Card	Apresentar a interface do jogo	Não
Mouse	Acertar em moles, clicar em botões, ver data e ver hora	Sim
RTC	Apresentar dia e hora no menu e no jogo. Guardar esta informação junto de cada score	Sim
Serial Port	Multiplayer game mode	Envio – Não (polling) Receção - Sim

2.1 Timer

O dispositivo timer assume um papel fundamental no desenvolvimento deste projeto, uma vez que o display do conteúdo do buffer gráfico é feito com base nas suas interrupções.

As interrupções do timer incrementam um contador e sempre que esse contador seja divisível por 5 (60/12) é atualizado o ecrã. Assim, a taxa de atualização do ecrã é de 30 vezes por segundo e é realizada com base na chamada da função *GeneralInterrupt* que terá como argumento o TIMER e o jogo atual.

Independentemente do estado do jogo atual optamos por redesenhar o ecrã na sua totalidade, ou seja, é desenhado o background do estado do jogo. Contudo conforme o estado atual do jogo serão desenhadas as *sprites* desejadas.

Em ambos os modos de jogo (*singleplayer* e *multiplayer*) o timer é utilizado para controlar o tempo de jogo incrementando um contador e verificando se este chegou ao valor da duração estipulada para o jogo. Assim que o jogo termina, o estado do jogo será *gameover*, o timer será usado para mover uma *sprite* de balões caso seja conquistado um novo *highscore*. No caso de o estado do jogo ser exit, o timer será utilizado para animar uma mole que se desloca no fundo do ecrã.

2.2 Keyboard

Ao longo do jogo, recorremos ao dispositivo *keyboard* para diversas situações diferentes. A cada interrupção do *keyboard*, a informação gerada é tratada no modo em que o utilizador se encontra caso este modo utilize a informação recebida (por exemplo no Menu Inicial o *keyboard* não tem funcionalidade pelo que as suas interrupções são ignoradas). Por um lado, quando o utilizador escolhe ver as instruções do jogo pode sair deste modo e regressar ao menu inicial pressionando a tecla ESC. Por outro lado, o *Player* poder definir o seu nome no modo *Player Settings* escolhendo as possíveis combinações de letras do teclado (Criamos o módulo *kbd_manager* que faz a atribuição de cada letra a um *scancode*). Por fim, é ainda possível utilizar o *keyboard* em modo de jogo sendo feita a seguinte atribuição de letras por toupeira: A S D (moles da linha inferior da esquerda para a direita) e J K L (moles da linha superior da esquerda para a direita).

2.3 Graphics Card

Para o projeto foi utilizado o modo de vídeo 0x115, 800x600. DOUBLE BUFFERING Para obter todas as imagens presentes no jogo, utilizamos XPM's algumas de uso grátis obtidas da internet e as restantes foram criadas por nós com recurso às aplicações: *Illustrator* (logo do jogo) e GIMP (para convert *png* em *xpm*, recortar e fazer as escalagem da imagem se pretendido). Recorremos também a XPMs para mostrar a data e a hora atuais obtidos através do RTC. As fontes usadas para texto e números são grátis. Para a transparência das XPMs é usada a cor de transparência do modo XPM_8_8_8_8. As funções relacionadas com a placa de vídeo encontram- se no ficheiro vd_card.

2.4 Mouse

O mouse é utilizado em todos os estados do jogo (exceto no modo *Instructions*), servindo sempre como cursor (ou para seleção de botões ou para jogar selecionando a mole que se pretende atingir). No início da função *game_main_loop* faz - se a subscrição das interrupções, sendo que o seu tratamento é feito pela função *mouse_ih()* do Módulo Mouse. Por outro lado, a função *mouse_get_event()* é definida no módulo do mouse e analisa o *packet* recebido pelo rato e gera um evento que depois será interpretado no *interrupt handler* do mouse de cada modo. Note-se que neste jogo o único evento a ter em conta é o LB_RELEASED, uma vez que a seleção de botões ou *holes* (no jogo) é feita carregando no botão esquerdo do rato. Devido à importância do cursor no nosso jogo criamos o módulo Cursor que contem a sua xpm, bem como posição. Aqui são definidas também as funções: *move_cursor()* - altera as coordenadas do cursor sempre que houver um movimento do rato (a posição atual do cursor é sempre obtida com base num movimento relativo à sua posição original) *draw_cursor()* - desenha o cursor na sua posição atual Por outro lado, definimos também as funções: *mouse_over()* que verifica se o cursor se encontra por cima do botão selecionado *check_over_mole()* que verifica se o martelo se encontra por cima da mole (na "cabeça" desta)

2.5 Rtc

O RTC é utilizado para apresentar a data e o tempo atual no menu principal, e para apresentar a data de um score na *leaderboard*.

A variável *rtc_date* é estática durante o funcionamento do programa, mas a variável *rtc_time* é atualizada a cada segundo através da utilização de interrupções de *update*. Estas variáveis são depois usadas para apresentar a data e tempo no menu principal, sempre atualizado, tal como para depois guardar a data de acontecimento de cada score. A função que trata das interrupções do RTC, *rtc_int_handler*, simplesmente lê para a variável *rtc_time*, o tempo atualizado assim que verificar que nenhum *update* está em curso.

A API criada para este dispositivo encontra-se no rtc.c e as contantes utilizadas encontram-se em rtc_macros.h.

2.6 Serial Port

O serial port é utilizado no modo *multiplayer* e tem como objetivo dar a possibilidade de dois computadores poderem comunicar o estado das toupeiras, se sobem, se são acertadas, tal como no final transmitirem entre si o resultado de cada *player*, anunciando o vencedor e perdedor.

São utilizadas subscrições do Received Data e Line Status e *polling* para a transmitir informação, pois dado a natureza do nosso jogo, em que apenas é transmitido um byte de cada vez, não existe necessidade de subscrever interrupções para a transmissão. A configuração estabelecida para o UART foi de 8 bits por *word*, com 2 bits de stop, sem paridade e com *bitrate* de 115200. No nosso caso, não demos diferença de performance para diferentes configurações.

Durante a subscrição do UART (COM1), *ser_subscribe_int*, é guardada a configuração inicial do UART (COM1) nas variáveis *initial_lcr, initial_ier, initial_dlm, initila_dll*, através da função *ser_save_init_conf*, que depois, durante a cancelação da subsrição, *ser_unsubscribe_int*, é restaurada pela função *ser_restore_init_conf*.

A função que trata das interrupções do UART (COM1), *ser_ih*, lê o IIR(Interrupt Identification Register), e caso tenha uma interrupção pendente, lê o byte presente no Receiver Buffer, caso haja uma *flag* de erro do Line Status Register (LSR) esteja ativa, é ativado um booleano a indicar que houve um erro na última leitura.

3. Organização e Estrutura do Código

3.1 Timer Module

Neste módulo estão presentes todas as funções criadas no âmbito do Lab2 para ser possível receber e utilizar interrupções do Timer.

Peso: 11%

Desenvolvido por todos os elementos do grupo.

3.2 Keyboard Module

Neste módulo estão presentes todas as funções criadas no âmbito do Lab3 para ser possível receber e utilizar interrupções do Keyboard.

Peso: 4%

Desenvolvido por todos os elementos do grupo.

3.3 Gaphics Card Module

Neste módulo estão presentes todas as funções criadas no âmbito do Lab5 para ser possível desenhar xpm's no ecrã bem como atualizar o mesmo sempre que desejado.

Peso: 6%

Desenvolvido por todos os elementos do grupo.

3.4 Mouse Module

Neste módulo estão presentes todas as funções criadas no âmbito do Lab4 para ser possível receber e utilizar interrupções do Mouse.

Peso: 8%

Desenvolvido por todos os elementos do grupo.

3.5 RTC Module

Neste módulo estão presentes todas as funções criadas para o tratamento do dispositivo RTC, permitindo receber interrupções e interpretar os seus registos e valores.

Peso: 4%

Desenvolvido pelo aluno Luís Matos.

3.6 Serial Port Module

Neste módulo estão presentes todas as funções criadas para o tratamento do dispositivo Serial Port, permitindo receber interrupções, interpretar os seus registos e valores, tal como configurar os atributos do UART.

Peso: 7%

Desenvolvido por todos os elementos do grupo.

3.8 Game module

Neste módulo está presente o ciclo de interrupções. Foi implementada uma maquina de estados para o estado do jogo, permitindo assim que consoante o estado do jogo atual será chamado o *interrupt handler* correspondente.

Peso: 18%

Desenvolvido por todos os elementos do grupo.

3.9 Menu module

Neste módulo é feito o *load* de todas as *sprites* envolventes o menu, bem como definidas as funções para desenhar as respetivas *sprites*. Este é o módulo corresponde ao estado inicial do jogo e conforme a seleção do utilizador o estado do jogo é alterado.

Peso: 7%

Desenvolvido pela aluna Sofia Germer.

3.10 Game-Over module

Neste módulo estão presentes todas as funções criadas mostrar a pontuação do jogador, sinalizar a sua adição à *leaderboard* e permitir sair do jogo, ver a *leaderboard* ou então voltar ao menu inicial para jogar de novo.

Peso: 5%

Desenvolvido pelo aluno Luís Matos.

3.11 Exit module

Neste módulo são definidas as funções que movem uma toupeira animada, mostram os créditos de jogo e fazem *free* de toda a memória alocada dinamicamente durante o *load* inicial do jogo.

Peso: 3%

Desenvolvido pela aluna Sofia Germer.

3.12 Player Settings module

Neste módulo são definidas três *structs*: *Player*, *Avatar* e *Player_Settings*, sendo definidas todas as funções de *load* e de *draw xpms*. Aqui são definidas as funções que permitem a escolha do martelo (*Avatar*) bem como a definição do nome do jogador.

Peso: 6%

Desenvolvido pelo aluno Pedro Machado.

3.13 Leaderboard module

Neste módulo definimos funções que gerem a informação relativa as pontuações obtidas no jogo. Para isso é definida uma função que lê informações relativas a pontuações de jogos anteriores, uma função que verifica se a pontuação obtida no último jogo é "merecedora" de um lugar no pódio, e uma função que atualiza o ficheiro com a nova pontuação. Para alem disso, tal como nos outros módulos, é feito o *load* das *sprites* relativas a ele e são definidas as funções que as desenham.

Peso: 6%

Desenvolvido pelo aluno Pedro Machado.

3.14 Mole module

Neste módulo estão presentes todas as funções relativas a uma toupeira. As três funções presentes neste módulo têm o papel de criar uma nova toupeira, modificar o estado em que uma toupeira se encontra (escondida, a subir, a descer), desenhar uma toupeira, respeitando sempre o seu estado atual e ainda verificar se o cursor se encontra sobre a cabeça de uma toupeira.

Peso: 9%

Desenvolvido pelos alunos Pedro Machado e Luís Matos.

3.15 Buttons module

Neste módulo é feito o load das sprites relativas aos botões e cada um é inicializado com o seu estado normal. Cada botão tem dois estados: o seu estado normal e o estado ativo, sendo que um botão é ativo quando o cursor se encontra por cima dele. São definidas, portanto, funções que atualizam e desenham os botões conforme o resultado da função, também deste módulo, que verifica se esse botão está ativo.

Peso: 5%

Desenvolvido pela aluna Sofia Germer.

3.16 Instructions module

Este módulo é bastante simples e consiste numa única *sprite* criada por nós com as regras do jogo. O utilizador apenas precisa de premir *ESC* quando pretender regressar ao menu principal.

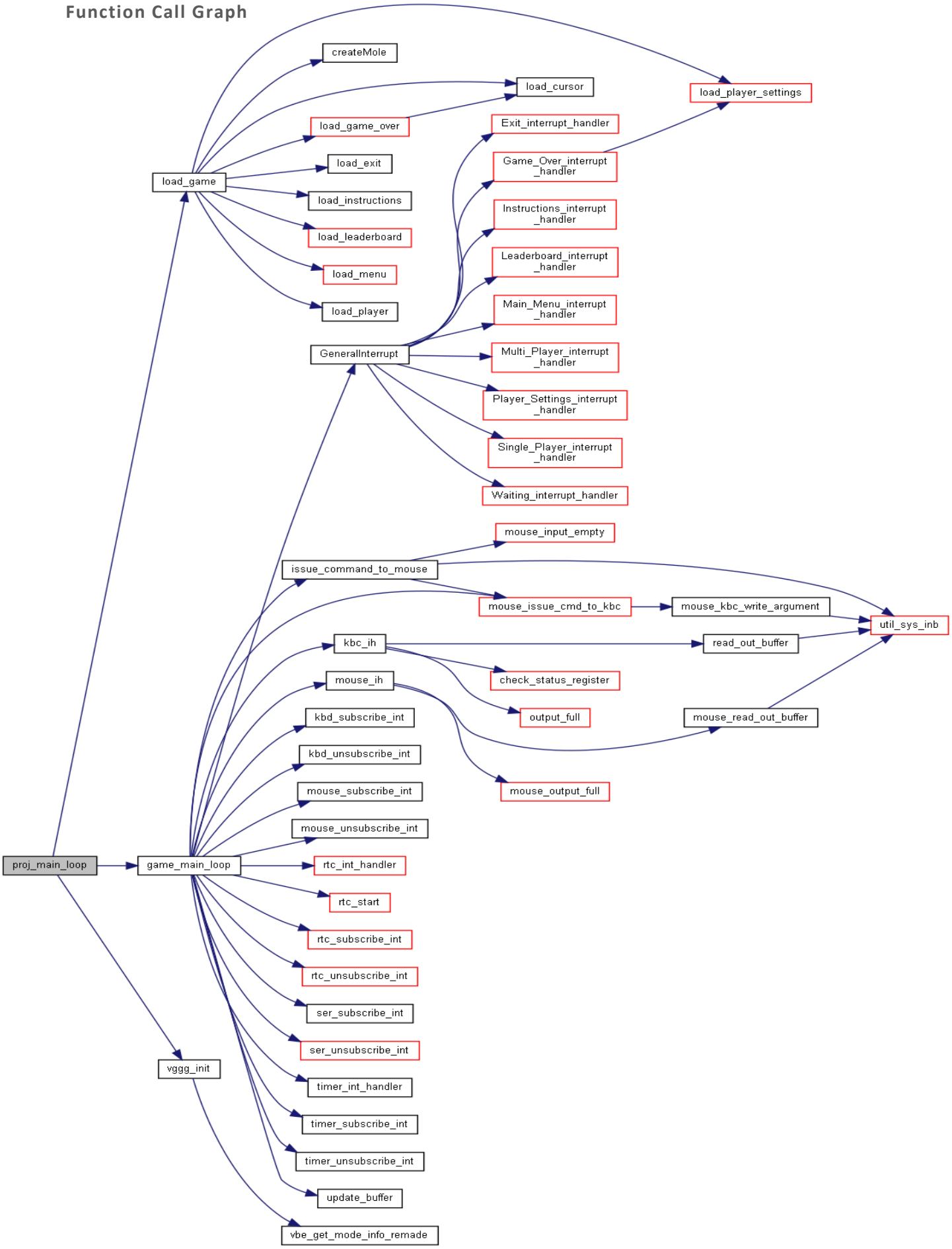
Peso: 1%

Desenvolvido pela aluna Sofia Germer.

3.17 Keyboard Manager Module

Neste módulo está presente uma só função que tem como objetivo traduzir um *scanCode* do *keyboard* na correspondente letra do abecedário. Optamos por criar este pequeno módulo auxiliar para não alterar o que importamos do Lab4, por uma questão de organização.

Function Call Graph



4. Detalhes de Implementação

Tendo como objetivo a melhor gestão possível das interrupções ao longo de todo o jogo, optamos por implementar uma maquina de estados que nos permite com base no estado do jogo em que nos encontramos lidar de forma diferente com cada interrupção recebida.

De forma a obter uma melhor fluidez de utilização do jogo decidimos fazer o *load* de todas as *xpm's* necessárias no início do jogo, bem como de todas as estruturas utilizadas ao longo do jogo (*leaderboard*, *player settings*, entre outras).

Ao nível dos botões que implementamos optamos por criar uma *struct* que guarda as informações sobre um botão e criar uma função para verificar se o cursor do rato se encontra por cima do botão, estas implementações permitiram uma melhor interação do jogador com o jogo, uma vez que sempre que o cursor se encontra sobre um botão a imagem do mesmo ficará mais clara.

Ao nível da escrita de números e letras no jogo foi criada uma função que recebe a fonte a utilizar, o valor/a palavra a desenhar e as suas coordenadas. Assim conseguimos utilizar uma única *xpm* para desenhar tudo o necessário. Além disso, foi ainda adicionado ao código relativo à placa gráfica providente do lab5 uma função que permite desenhar apenas parte de um *xpm* (ex: apenas uma letra/número).

Para obter o desenho de uma toupeira tiramos vantagem da implementação de uma *enum* que guarda as suas posições, uma vez que a cada posição é possível associar uma *sprite* previamente guardada num *array* de *sprites* associado a cada toupeira. Assim, para animar uma toupeira apenas necessitamos de alterar a sua *position* o que fará com que seja desenhada uma imagem diferente. Desta forma a posição da toupeira ao longo de todo o jogo é imutável, sendo apenas a sua imagem que muda.

O RTC foi implementado em conjunto com duas estruturas de dados auxiliares Date e Time que guardam o valor da data e hora atuais. Inicialmente, tal como definido na especificação, tínhamos planeado utilizar o RTC para alternar entre modo diurno e noturno. Contudo, optamos por utiliza-lo para mostra a data e hora atuais no menu inicial e para guardar informação sobre a data em que um *score* foi obtido. Consideramos que a nova funcionalidade é mais proveitosa e gera uma melhor interação com o jogo.

Implementamos também a possibilidade de jogar o jogo várias vezes, ou seja, no final de um jogo o jogador apenas tem de regressar ao menu principal e poderá jogar de novo. Para tal necessitamos de criar uma função que é responsável por fazer o *reset* a todas as moles, colocando a sua *position* como *hided* e o seu *time up* a zero.

Ao nível do modo de jogo *multiplayer* tínhamos que seria um modo cooperativo, contudo acabamos por alterar para um modo competitivo. Julgamos que este modo é mais desafiador ao nível da sua implementação uma vez que é necessário ter informação acerca de quem venceu, informação essa que não seria necessária no modo cooperativo.

Para ser possível a *leaderboard* estar sempre atualizada e presente em todas as execuções do programa, foi necessário ler e escrever um ficheiro, leaderboard.txt. Trata-se de um ficheiro de texto simples que inclui 3 linhas por cada score, um para o nome do *player*, outra para o score, e a última para a data da realização deste score, ordenados por melhor pontuação. Caso ainda não se tenham preenchido todas as posições da *leaderboard*, os scores sem valor têm como autor "*nobody*", e servem para "avisar" o programa que se trata de um score inútil, que não deve ser mostrado na *leaderboard*. As funções utilizadas na leitura e escrita deste ficheiro são *load_scores* e *save_scores*, respetivamente, presentes no ficheiro score.c.

```
PEDRO    <- nome
100      <- score
2/1/21   <- data
SOFIA
90
1/1/21
LUIS
75
3/1/21
JOAO
50
4/1/21
nobody   <- score por preencher
0
0/0/0
```

5. Conclusões

Após terminado este projeto podemos, sem dúvida alguma, concluir que esta unidade curricular foi extremamente desafiante o que culminou na obtenção de novas competências no que diz respeito à programação de baixo nível.

Foi de facto um desafio desenvolver este jogo, não só pela dificuldade intrínseca que um projeto destes contêm, mas também pela necessidade de uma gestão de tempo muito cuidada, para que as outras unidades curriculares e os seus trabalhos não fossem comprometidos.

Salientamos que o modo como foram realizadas as aulas práticas, 15 em 15 dias com apenas metade da turma presente, apesar de muito vantajoso e importante ao nível da saúde, contribuiu negativamente para a aprendizagem. De facto, não só pela menor, ou quase nula, interação com outros colegas de turma, mas também pela menor quantidade de aulas, algumas dificuldades acabaram por se manifestar de forma mais expressiva. Além disso, o facto de não trabalharmos de forma presencial todos juntos foi também um fator negativo.

A opção que tomamos de incluir no nosso jogo os dispositivos complementares RTC e Serial Port foi, de facto, importante na estimulação da autoaprendizagem, uma habilidade importantíssima na área da programação, quer de alto como de baixo nível. Apesar de serem dispositivos complementares, as informações disponibilizadas sobre a sua implementação e ainda a possibilidade testar a nossa implementação (recorrendo ao Lab 6 e 7), condicionou muito positivamente a possibilidade de estarem presentes no nosso jogo.