DRAW IT

UM JOGO DE DESENHO

Mestrado Integrado em Engenharia Informática e Computação

Laboratório de computadores - Turma 1 Grupo 9

Miguel Boaventura Rodrigues (up201906042@fe.up.pt)

Tiago Caldas da Silva (up201906045@fe.up.pt)

Índice

Instruções de utilização do programa 3
Menu Principal 3
Menu das definições/opções 4
Menu *Playground* 4
Menu de seleção dos avatares 5
Menu de jogo 6
Estado do Projeto 9
Dispositivos Utilizados 9
Descrição das funcionalidades dos diferentes dispositivos 9
Timer 9
Teclado 9
Rato 10
Placa de vídeo 10
RTC (*Real Time Clock*) 10
Estrutura e modulação do código 11
Módulo Timer 11
Módulo Teclado 11
Módulo Rato 11
Módulo Placa de Vídeo 12
Módulo RTC (*Real Time Clock*) 12
Módulo das *Sprites* 13
Módulo do *Canvas* 13
Módulo do Avatar 13
Módulo do *Playground* e Ecrã de Jogo 13
Módulo das Definições 14
Módulo da Lógica (Máquinas de Estado) 14
Relatório e Documentação 14
Detalhes de Implementação 16
Conclusões 18
Apêndice 19

Índice de Figuras

Figura 1 - Menu Inicial 3
Figura 2 - Menu das definições/opções 4
Figura 3 – Ecrã do *Playground* 4
Figura 4 - Menu de seleção dos avatares no estado inicial 5
Figura 5 - Menu de seleção dos avatares após a escolha do primeiro jogador. 5
Figura 6 - Estado inicial do menu de jogo 6
Figura 7 - *Blocker* no ecrã de jogo 6
Figura 8 - Opções disponíveis no menu deslizante 7
Figura 9 - Jogo em pausa 7
Figura 10 - Mensagem dirigida ao vencedor do jogo 8
Figura 11 - Ecrãs para o palpite sobre aquilo que foi desenhado no menu de jogo 8
Figura 12 - *Sprites* que compõe a animação do relógio 12
Figura 13 - Gráfico das Chamadas efetuadas pela função proj_main_loop() no ficheiro proj.c 15

Índice de Tabelas

Tabela 1 - Dispositivos utilizados e a respetiva funcionalidade 9
Tabela 2 - Distribuição dos Módulos 14

Instruções de utilização do programa

O DRAW IT pretende ser um jogo que segue os mesmos traços de um jogo bastante popular na internet conhecido como “skribbl.io”.

O jogo divide-se em vários turnos onde um dos jogadores desenha de acordo com um mote previamente selecionado pelo próprio. De seguida, cabe ao outro jogador adivinhar aquilo que foi desenhado. O jogo termina quando o número máximo de rondas é atingido – esta opção não é dada ao utilizador. Terminados todos os turnos, o vencedor é aquele com o maior número de pontos. Se o número de pontos for o mesmo, então haverá mais rondas até que se apure um vencedor.

Um jogo simples, mas muito divertido. Foi este o motivo que levou à nossa escolha!

Menu Principal

Assim que o programa começa, o utilizador confrontar-se-á com um menu relativamente simples, onde dispõe de 3 opções. Para prosseguir, o jogador deve utilizar o rato e clicar sobre o botão da opção que deseja.

Figura 1 - Menu Inicial

Como é possível observar, o utilizador tem uma opção que o reencaminha para o ecrã de seleção dos avatares (mostrado mais à frente) – a opção *PLAY*. Na segunda opção, o jogador é levado ao ecrã das definições (também referenciado mais abaixo). Por fim, a última opção *EXIT* é autoexplicativa – leva ao fim da execução do programa.

Menu das definições/opções

Neste menu, o utilizador pode ajustar as definições do jogo ao seu gosto, como por exemplo, o tempo de jogo. Pode ainda também aceder ao *Playground*, algo que será abordado mais à frente.

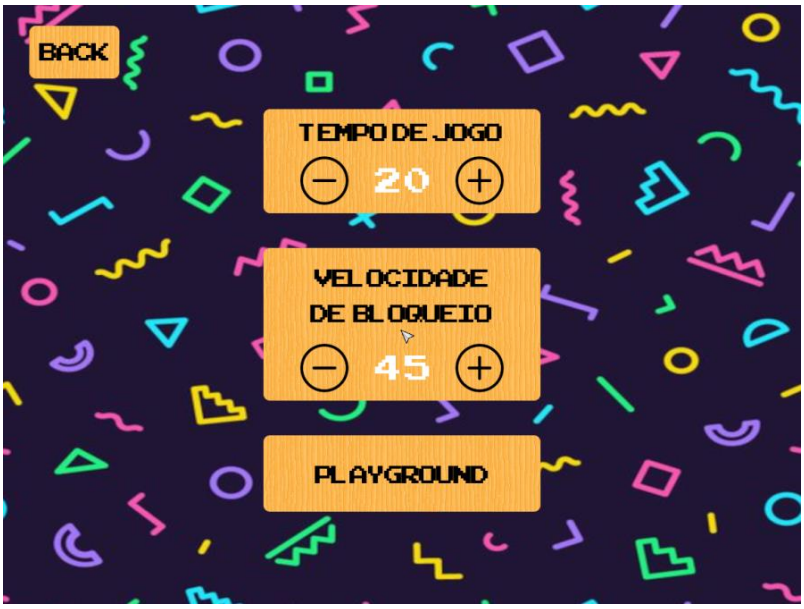

Figura 2 - Menu das definições/opções

Menu *Playground*

Este menu pode ser acedido através do menu das definições. Aqui é dada ao utilizador a possibilidade de expressar os seus dotes artísticos e criativos! Neste menu, o jogador pode, porventura, ambientar-se com o menu de jogo que é muito similar àquele encontrado no ecrã de jogo - que veremos mais adiante.

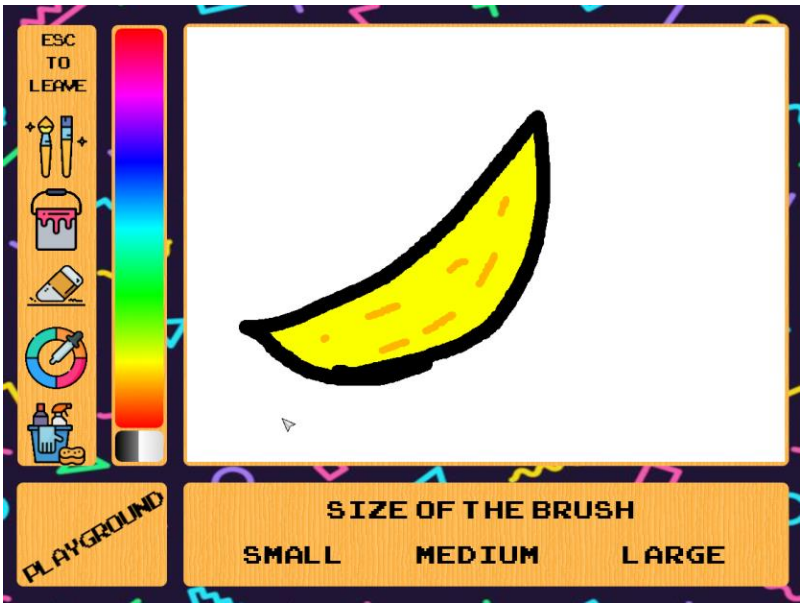

Figura 3 – Ecrã do Playground

Menu de seleção dos avatares

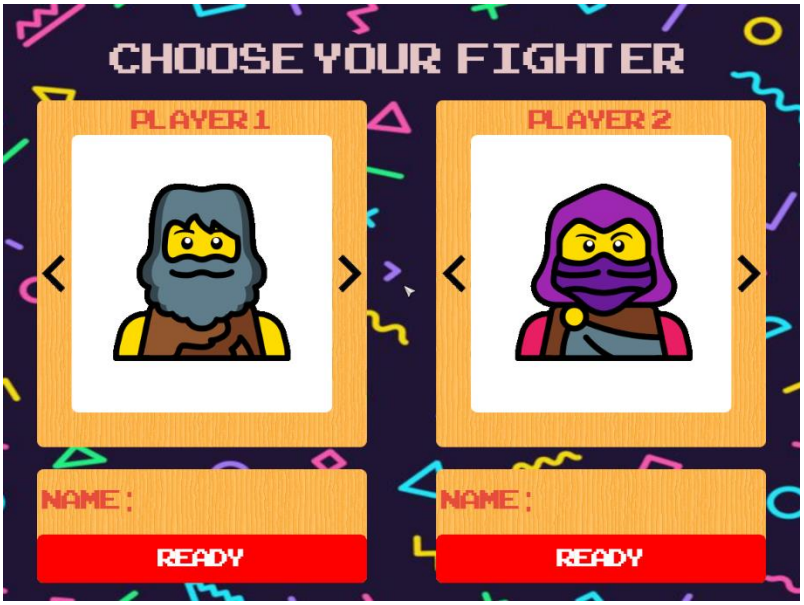

Figura 4 - Menu de seleção dos avatares no estado inicial

Neste menu, ambos os jogadores são conduzidos à escolha dos seus avatares e respetivos *nicknames*, um de cada vez. Ou seja, uma vez que o primeiro jogador escolheu o seu avatar e inseriu o seu nome usando o teclado, o próximo jogador escolhe o seu avatar e insere o seu nome.

Após a escolha por parte do segundo jogador, o programa segue para o ecrã de jogo.

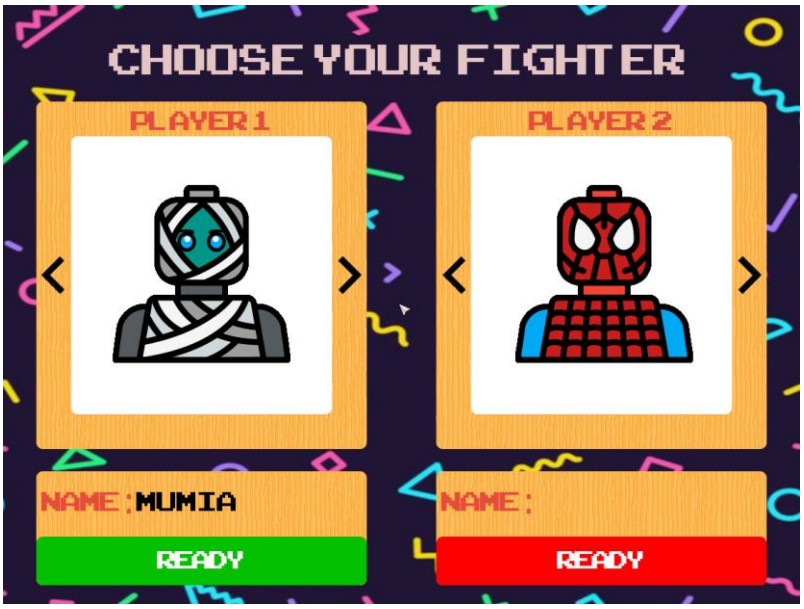

Figura 5 - Menu de seleção dos avatares após a escolha do primeiro jogador.

Menu de jogo

É neste menu onde se encontra a esmagadora maioria da lógica por detrás daquilo que é este projeto.

Figura 6 - Estado inicial do menu de jogo

Como é possível observar pela figura, o utilizador que irá desenhar deve pressionar com o botão esquerdo do rato a palavra que deseja desenhar. Também é possível notar um quadro de pontuações no canto inferior esquerdo, bem como dois submenus – para a paleta de cores e para as diferentes ferramentas de pintura.

Passando agora, para aquilo que é o núcleo do jogo, no ecrã irá surgir um retângulo ao qual decidimos chamar de *blocker*. O propósito deste objeto no ecrã é bloquear uma porção da tela impedindo que o jogador desenhe nessa área.

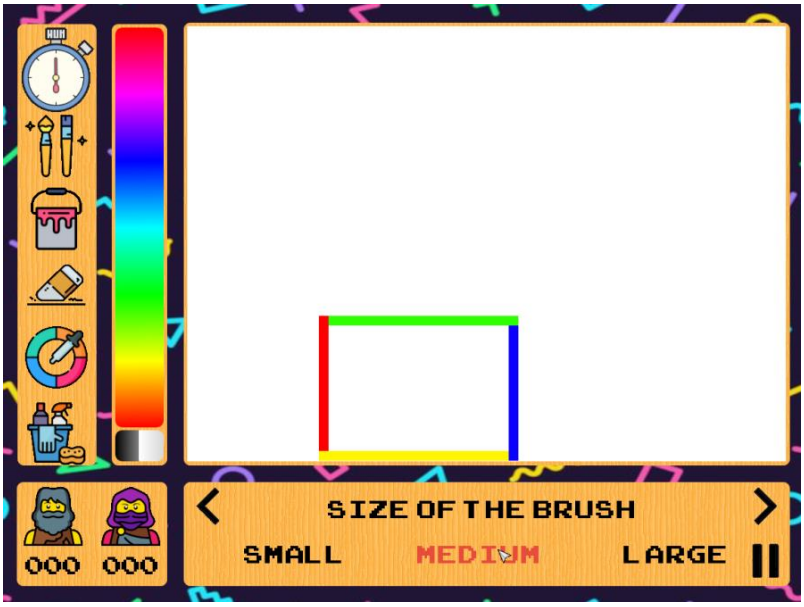

Figura 7 - Blocker no ecrã de jogo

O ecrã de jogo possui ainda um menu deslizante para acomodar as várias opções, localizado no fundo. Neste menu, é possível colocar o jogo em pausa, desistir da jogada em curso ou até mesmo dar por terminado o desenho.

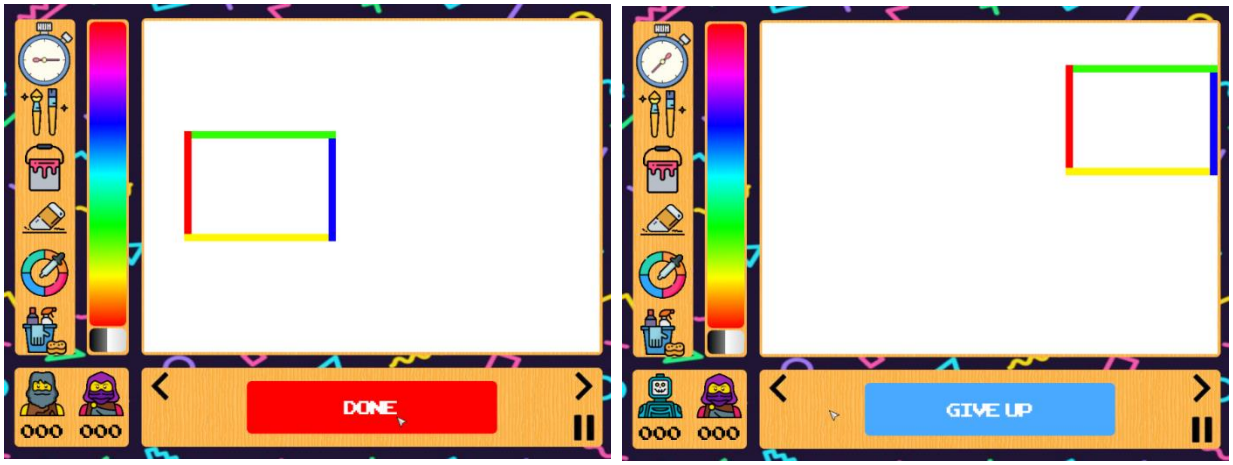

Figura 8 - Opções disponíveis no menu deslizante

Figura 9 - Jogo em pausa

Após o término de uma ronda, o jogador que não pintou é convidado a fazer um palpite sobre aquilo que foi desenhado. Nos ecrãs acima é possível constatar as diferenças entre um palpite certo e um palpite errado. Outro aspeto importante, é que o jogador que está a adivinhar a palavra tem o mesmo tempo para pintar quanto o outro jogador.

Figura 11 - Ecrãs para o palpite sobre aquilo que foi desenhado no menu de jogo

Finalmente, quando se der o caso de todos turnos terem terminado e houver um vencedor apurado, o ecrã de jogo apresentará uma mensagem customizada a congratular o vencedor com o nome que inseriu no menu de seleção do avatar.

Figura 10 - Mensagem dirigida ao vencedor do jogo

Estado do Projeto

Dispositivos Utilizados

Dispositivo	Utilização	Interrupções
Timer	Garantir a atualização da lógica do jogo e da informação apresentada no ecrã de uma forma constante no tempo.	Sim
Teclado	Permitir a inserção dos nomes dos jogadores e do palpite sobre o desenho depois de decorrido o tempo previamente estipulado.	Sim
Rato	Interação do jogador com os diferentes elementos presentes no ecrã.	Sim
Placa de vídeo	Apresentação da interface gráfica, indispensável no caso deste projeto.	N/A
RTC	Usado para a medição do tempo decorrido numa determinada jogada	Sim (Alarmes e Atualização)
Porta Série	A porta série não foi implementada devido aos constrangimentos que atrasaram o desenvolvimento.	N/A

Tabela 1 - Dispositivos utilizados e a respetiva funcionalidade

Descrição das funcionalidades dos diferentes dispositivos

Timer

O *timer* foi o primeiro dispositivo a ser implementado e o seu propósito é garantir que a execução do programa permanece constante ao longo do tempo. Em primeiro lugar, são subscritas as suas interrupções usando para isso a função *timer_subscribe_int()*; de seguida, é definida a taxa de atualização do ecrã com a função *timer_set_frequency()*.

A cada interrupção do timer é atualizado o ecrã, sendo que o mecanismo utilizado foi o de efetuar a cópia dos elementos presentes no *buffer* secundário para o *buffer* principal – ver secção da placa gráfica.

Teclado

O principal propósito do teclado serve para os jogadores inserirem os seus nomes e também para que depois do desenho um dos jogadores possa escrever o seu palpite. A cada interrupção do teclado e, se no instante de execução assim se proporcionar, é pedido ao jogador que insira algum tipo de dado, estes são escritos para um *array* para serem posteriormente apresentados no ecrã – abordado mais à frente.

Rato

O rato é sem dúvida um dispositivo que facilita as interações entre a máquina e o utilizador. O rato serve como o pincel da tela, onde o utilizador pode pintar. Este dispositivo também é responsável pela interação entre os menus. A sua posição interage muito com o fluxo de execução, na medida em que existem imensas animações de *sprites* ativas quando o rato se encontra por cima destes.

Placa de vídeo

A placa de vídeo foi utilizada para apresentar a interface aos jogadores. Como no nosso jogo o domínio visual prevalece sobre os restantes, optamos por usar o modo gráfico 0x14C. Este modo permite representar cerca de 16.8 milhões de cores diferentes[1] e cada pixel é composto por 4 bytes – onde o byte mais significativo remete para a transparência da cor e os restantes para os canais RGB. Desta maneira, o jogador pode escolher qualquer cor à sua escolha para o seu desenho. Por último, este modo gráfico tem a resolução de 1152 pixéis horizontais por 864 pixéis verticais (1152 x 864).

Também é de salientar que optamos por implementar a técnica de *double buffering* com cópia, isto é, toda a informação é escrita no *buffer* secundário até que, quando ocorre uma interrupção do *timer*, toda a sua informação é copiada com o auxílio da função da biblioteca do C – *memcpy()*.

RTC (*Real Time Clock*)

A utilidade do RTC fundamentou-se essencialmente na medição do tempo. Em primeiro lugar, as suas interrupções são subscritas usando a função *rtc_subscribe_int()*. Mais tarde, no início do jogo, a animação do relógio no menu do jogo ocorre a par com as interrupções de *update* – geradas a cada segundo. Da mesma forma, quando o tempo para desenhar se esgota, é gerada uma interrupção do tipo alarme.

[1] 16 777 216 ou 2^{24} cores.

Estrutura e modulação do código

Módulo *Timer*

O módulo do *timer* foi desenvolvido durante as aulas do *lab2*. Como foi dito anteriormente, são as componentes deste módulo que asseguram que o jogo executa com a taxa de atualização estipulada.

A taxa de atualização é definida com uma chamada à função *timer_set_frequency()*. Este módulo também é composto pelas funções que permitem ativar e/ou desativar a linha de interrupção deste dispositivo (função *timer_subscribe_int()* e função *timer_unsubscribe_int()*, respetivamente).

Este módulo teve um peso de 2.5%., tendo sido desenvolvido por ambos os elementos do grupo.

Módulo Teclado

O módulo do teclado foi maioritariamente desenvolvido durante o *lab3*, contudo, foi adaptado às necessidades do projeto.

Este módulo possui como principais funções aquelas que ativam e desativam as interrupções do teclado (função *keyboard_subscribe_int()* e função *keyboard_unsubscribe_int()*, respetivamente), mas também a função *keyboard_recognizeScanCode()* que, como o próprio nome o indica, transforma os *scancodes* gerados pelo dispositivo em caracteres legíveis. Também é dada a possibilidade de usar o teclado como dispositivo de navegação entre menus, para além do rato.

Este módulo teve um peso de 5%, tendo sido desenvolvido por ambos os elementos do grupo.

Módulo Rato

O módulo do rato foi, tal como os anteriores, desenvolvido na sua maioria durante as aulas do *lab4*, mas uma vez mais teve de ser ajustado ao propósito do projeto. Este módulo é, na verdade, o dispositivo mais relevante e com o qual o utilizador mais interage. O rato é usado para a navegação entre os diferentes ecrãs e/ou menus. Para além disso, funciona como pincel quando o seu botão esquerdo se encontra pressionado e a sua posição dentro da tela do ecrã de jogo.

Este módulo tem como principal função a função *mouse_send_command()*, que envia o comando *enable_data_reporting* de modo a permitir o envio de pacotes em *Stream Mode*. Também as habituais funções de (des)ativação da interrupções - *mouse_subscribe_int()* e *mouse_unsubscribe_int()*. E por último, a função *move_cursor()*, responsável pelo deslocamento e desenho do *xpm* sobre a **struct** de s*prite* associada ao rato.

Este módulo teve um peso de 7.5%, tendo sido desenvolvido pelos dois elementos do grupo.

Módulo Placa de Vídeo

A placa de vídeo, ou placa gráfica, é a responsável pela apresentação de todos os elementos no ecrã, como por exemplo as *sprites*. Este módulo foi desenvolvido durante o *lab5* e revelou-se muito importante no projeto, visto que o seu tema está muito ligado ao apelo visual.

O módulo tem como principais funções: a função *video_card_init()* – que inicia a placa no modo 0x14C (referido em detalhe na funcionalidade do dispositivo associado ao módulo) e a função *video_card_get_info()* – que efetua a chamada à função 00h da norma *vbe* e preenche a **struct** vbe_info com todos os dados necessários relativos ao modo gráfico em uso. As funções *video_card_set_pixel()* e *video_card_get_pixel()* pintam com uma determinada cor ou retornam a cor de um determinado pixel, respetivamente, e também merecem a sua importância, isto porque são usadas noutros módulos com alguma frequência. Referir também a tentativa de uso de *page flipping* com a função *video_card_page_flip()*.

Este módulo teve um peso de 2.5%, tendo sido repartido pelos dois elementos do grupo.

Módulo RTC (*Real Time Clock*)

O RTC foi um dispositivo relativamente simples de ser implementado depois de se conhecer as suas caraterísticas. O dispositivo é usado para duas funções: a primeira, provavelmente mais relevante, é determinar quando o tempo de desenho deve terminar, gerando com isso um alarme; a segunda é manter a contagem do tempo para que a *sprite* do relógio, presente no canto superior esquerdo do menu de jogo, seja animada.

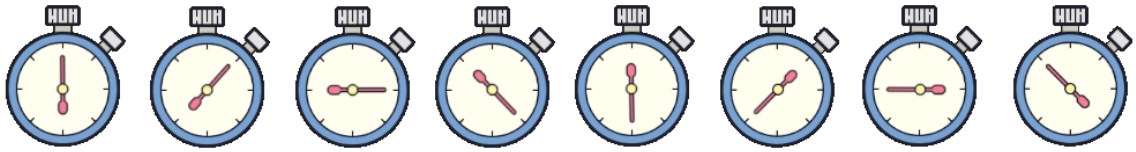

Figura 12 - Sprites que compõe a animação do relógio

Este módulo tem várias funções com importância relativamente semelhante, no entanto ficam aqui algumas. Primeiro, as funções *rtc_enable_conf()* e *rtc_disable_conf()*, funções que recebem como parâmetro um tipo enumerado que ativa ou desativa o bit correspondente à interrupção no registo B. Também referir a importância das funções *rtc_read_value()* e *rtc_write_value()* que, como os próprios nomes indicam, efetuam a escrita e a leitura para um dos registos que armazenam os valores de tempo. Para este mecanismo, foi usado também um enumerado (**rtc_time_value_t**) que passa qual o registo que deve ser acedido.

Este módulo teve um peso de 5%. O desenvolvimento do módulo ficou a cargo do Miguel Rodrigues.

Módulo das *Sprites*

Este módulo representa todas as funções que possuem manipulação de *sprites*. As principais funções são aquelas mais utilizadas ao longo de todo o projeto: a função *draw_sprite()* e a função *erase_sprite()*.

O peso deste módulo rondou os 5%. Ambos os elementos participaram no desenvolvimento deste módulo.

Módulo do *Canvas*

Este módulo é relativamente simples e nele estão contidas as funções que permitem que o ato de desenhar ocorra, de uma forma semelhante, a programas de desenho como *MS Paint*.

Neste módulo, é de destacar as funções *canvas_fill_brush()* e *canvas_draw_brush()* que em conjunto permitem simular um pincel (seguindo o deslocamento do rato), com um tamanho e cor próprio. De referir também a função *canvas_fill_bucket()* que reproduz o balde de tinta – preenchendo uma área com uma determinada cor. De outra forma, a função *canvas_cleaning()* limpa todo o *canvas* assim que o respetivo botão é premido.

Este módulo teve um peso de 15%. O desenvolvimento do módulo ficou a cargo do Miguel Rodrigues, contudo ambos os elementos contribuíram.

Módulo do Avatar

No módulo do avatar englobam-se funções *move_avatar()*, que é a responsável pela animação de seleção do avatar. Neste módulo englobam-se funções como a *draw_name_player1()* e draw*_name_player2()* que desenham o nome dos respetivos jogadores no ecrã de seleção dos avatares. Estas funções são chamadas pelo mecanismo de apontadores para funções, um tópico abordado nos detalhes de implementação.

Este módulo teve um peso de 15%. Este módulo foi inteiramente desenvolvido pelo Tiago Silva.

Módulo do *Playground* e Ecrã de Jogo

Como podemos ver anteriormente, o ecrã principal de jogo - com dois jogadores - e o *Playground* são muito idênticos, sendo, na realidade, um deles a adaptação do outro.

Este módulo refere-se aos ecrãs onde é possível pintar, estando ambos associados ao módulo do *Canvas*, na medida em que este é, essencialmente, uma dependência necessária ao seu propósito. Desta forma, este módulo merece um certo relevo ou destaque, na medida em que possui funções como a função *play_menu_state_machine(),* que possuem mecanismos para calcular onde as *sprites* devem ser colocadas tendo em conta diferentes parâmetros.

Este módulo teve um peso de 5%. Este módulo ficou a cargo do Tiago Silva, todavia ambos os elementos contribuíram.

Módulo das Definições

Neste módulo, estão presentes as funções relacionadas com o ecrã das definições. Destacam-se as funções *function_time()* e *function_velocity()*, que coordenam os valores das variáveis do tempo e da velocidade de jogo com aquilo que é apresentado no ecrã.

Este módulo teve um peso de 7.5%, tendo sido desenvolvido pelo Tiago Silva.

Módulo da Lógica (Máquinas de Estado)

Este é, por uma larga margem, a componente do projeto mais expressiva e mais dispendiosa no que toca a tempo. Aqui entram vários tópicos, como as máquinas de estados, essenciais na programação com base em eventos – como a escolha de uma determinada opção no menu inicial. De facto, a presença deste módulo ao longo de todo o projeto é notável, considerando que o programa possui bastantes menus e dá ao utilizador bastantes opções.

Este módulo teve um peso de 30%. Foi maioritariamente desenvolvido pelo Tiago Silva. Não obstante, os dois elementos participaram no seu desenvolvimento.

Módulo	**Peso do Módulo**	**Contribuição**
Timer	2.5 %	Ambos
Teclado	5 %	Ambos
Rato	7.5 %	Ambos
Placa de Vídeo	2.5 %	Ambos
RTC	5 %	Miguel Rodrigues
Sprites	5 %	Ambos
Canvas	15 %	Ambos
Avatar	15 %	Tiago Silva
Playground/Jogo	5 %	Ambos
Definições	7.5 %	Tiago Silva
Lógica (Máquinas de Estado)	30 %	Ambos

Tabela 2 - Distribuição dos Módulos

Relatório e Documentação

Relativamente à produção deste relatório e à geração da documentação necessária à entrega deste projeto, ambas as tarefas ficaram a cargo do Miguel Rodrigues.

Gráfico das chamadas das funções

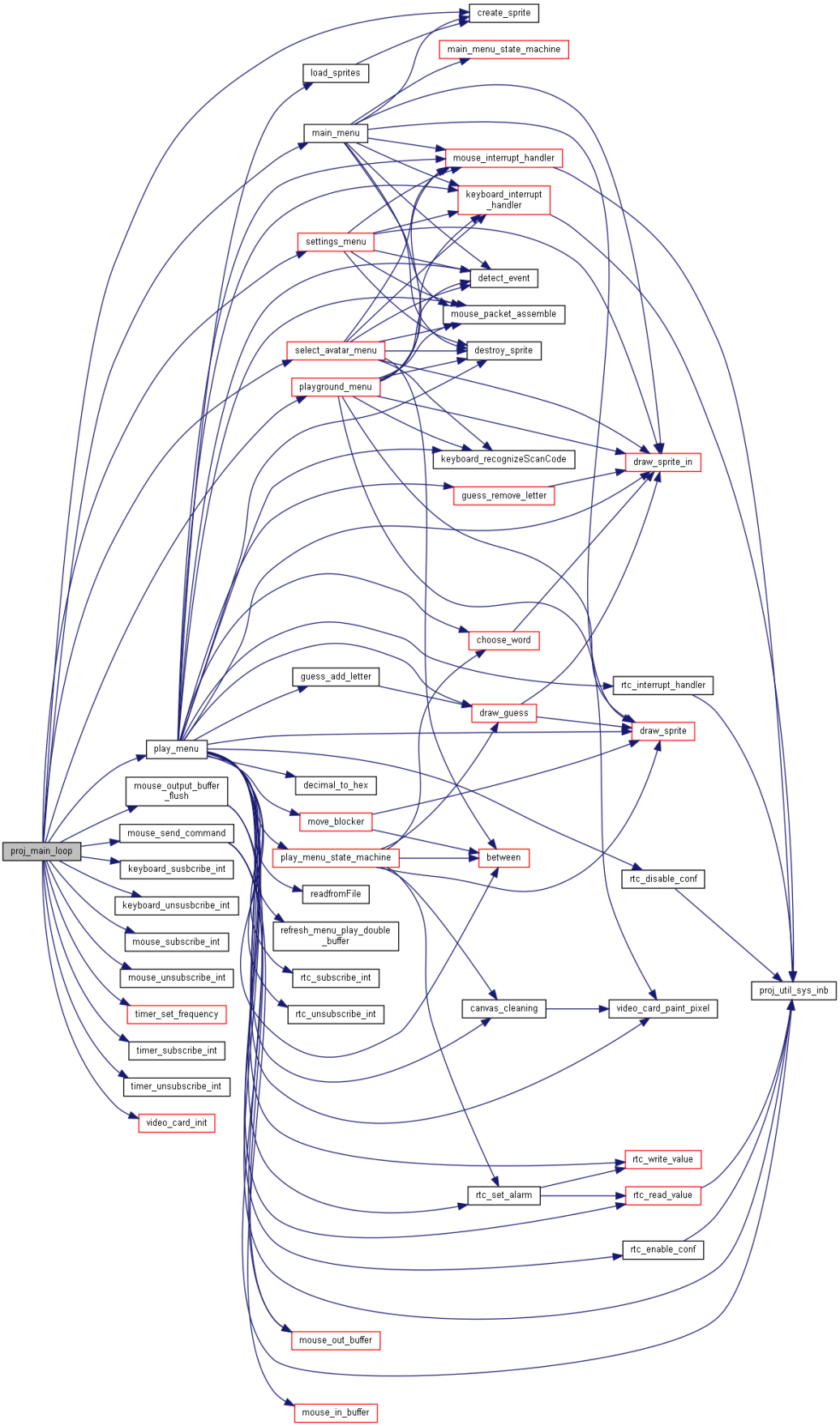

Figura 13 - Gráfico das Chamadas efetuadas pela função proj_main_loop() no ficheiro proj.c

Detalhes de Implementação

No nosso caso em particular, quando o teclado recebe uma interrupção e nesse mesmo instante é necessário que o utilizador insira alguns dados (e.g. nome do jogador no ecrã dos avatares), então o *scancode* que gerou essa interrupção será convertido num caractere a ser escrito num *array* para mais tarde poder ser apresentado no ecrã – na seguinte interrupção do timer. O mesmo mecanismo é usado mais à frente, na fase do palpite. A função que está na base de todo este mecanismo, e já referida anteriormente, é a função *keyboard_recognizeScanCode()*.

Na placa de vídeo, usamos as funções 01h e 02h do *standard* VBE. A primeira é usada na função *video_card_get_info()* e a segunda na função *video_card_init()*. Neste projeto, também tentamos implementar *page flipping* – tirando proveito da função 07h do *standard* VBE – no entanto não fomos bem-sucedidos, pelo que adotamos a técnica de *double buffering* com a cópia, ou seja, alocamos dinamicamente um *buffer* de tamanho igual ao do ecrã, onde eram desenhados todos os elementos, para que na interrupção seguinte do *timer* fossem apresentados no ecrã principal. Nessa atualização, é usada a função *memcpy()*, definida na biblioteca principal da linguagem C, para a cópia dos valores entre os *buffers*.

Relativamente à medição do tempo com o RTC, esta funcionalidade poderia ter sido implementada no módulo *timer*, mas preferimos fazer o uso das interrupções de *update* caraterísticas do dispositivo, visto que estas são geradas quando um dos registos efetua uma mudança no seu valor, ou seja, de segundo a segundo. Ainda relativamente ao *handler* do RTC criamos um mecanismo de identificação das interrupções com base em manipulação de bits.

Uma das componentes mais notáveis no nosso programa é o uso de *sprites* e de movimento, aliás, sempre que há um movimento, por exemplo, de um avatar no ecrã de seleção dos avatares, o restante programa "suspende" momentaneamente a sua execução para dar ênfase ao movimento. Por falar em movimento, no ecrã de jogo decidimos implementar um *blocker* com esta caraterística, uma vez que, na nossa opinião, o jogo ainda estava demasiado estático. Também é de salientar a existência de um *array* de tamanho equivalente ao do *pixmap* de uma determinada *sprite* para que o rasto deixado durante o movimento seja transparente e não haja artefactos indesejados.

No ecrã de escolha dos avatares, é dada ao utilizador a possibilidade de escolher um personagem diferente daquele que é pré-definido. Desse modo, irá ocorrer uma animação onde é necessário verificar a colisão entre *sprites* - do *sprite* correspondente à imagem do jogador com o *sprite* do painel de fundo, local até onde o primeiro deve ser mostrado.

No módulo do *Canvas*, é dada a possibilidade ao utilizador de usar diversas ferramentas para pintar, por exemplo, o utilizador pode escolher o tamanho do pincel ou então usar o balde de tinta. O acompanhamento daquilo que o utilizador está a utilizar foi implementado por tipos enumerados declarados no ficheiro *canvas.h*.

Ainda neste módulo, é importante referir que a função *canvas_bucket_fill()* usa o algoritmo *flood fill*. O nosso código foi adaptado a partir daquele que se encontra em https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint/. Este algoritmo apresenta inúmeras aplicações nos mais diversos contextos. Na verdade, este algoritmo pode ser implementado sob várias abordagens, todavia optamos por uma abordagem mais simples: a que utiliza recursividade.

Outro problema que enfrentamos, foi o facto de os conteúdos do canvas serem apagados a cada atualização do ecrã. Desse modo, a nossa solução passou por adotar a alocação de um terceiro

buffer auxiliar, para que esses elementos permanecessem em memória, mesmo entre *frames* consecutivos.

Já na parte da lógica, é de salientar o uso de máquinas de estado com base em tipos enumerados, sendo um exemplo o **enum** *mouse_event* usado para detetar os diferentes tipos de eventos que possam acontecer com a utilização do rato. Em todos os menus existem máquinas de estado que permitem coordenar a execução do programa com os eventos que ocorreram. Nestas máquinas de estado há o uso de *arrays* de apontadores para funções. Observamos que esta funcionalidade da linguagem C permite escrever código de uma forma mais concisa, e que se trata de uma caraterística muito poderosa e com imensos exemplos práticos de utilização.

No módulo da lógica, resolvemos incluir ainda a manipulação de ficheiros de texto. Estes são responsáveis por armazenar as várias palavras para que o jogador possa desenhar. As funções *readFromFile()* e *chooseWord()* garantem que esta escolha é feita de forma aleatória.

Conclusões

Chegados ao ponto sobre o qual somos chamados a refletir sobre a unidade curricular. Não podemos deixar de salientar que existem pontos muito positivos, bem como áreas que carecem de alterações.

Em retrospetiva, ambos os elementos do grupo consideram que se trata de uma UC com conteúdos interessantes e que, se explorados de forma apelativa, abrem muitos caminhos para aqueles que gostam de programação de baixo-nível e de periféricos.

Temos consciência que, neste ano letivo em particular, tudo ocorreu sob condições adversas: a impossibilidade de estar presente em todas as aulas práticas conjugado com um calendário muito pouco favorável (2 feriados do mês de dezembro) não permitiram esclarecer muitas dúvidas – que, por vezes, são difíceis de ser expostas.

Para além disso, existem aspetos menos positivos e que muitas vezes atiram os alunos para situações extremamente delicadas. Em primeiro lugar, um dos problemas que consideramos que seja rapidamente endereçado, é a falta de feedback em relação ao código desenvolvido nos *labs*, o que influenciou muito a estruturação do projeto. Também de referir a demorada correção das provas práticas, que também não ajudou os alunos.

Da mesma maneira, a LCF demonstrou, por motivos até à data desconhecidos, ser muito instável. No nosso caso, foi de tal ordem grave, que ficamos impossibilitados de implementar todos os dispositivos propostos (porta-série), muito graças ao tempo despendido na tentativa da resolução de erros que estavam para lá do nosso alcance.

Aproveitamos ainda para dizer que certas componentes letivas mereciam uma atenção mais pormenorizada, visto que é notável um choque entre aquilo que foi ensinado em anos anteriores e aquilo que é proposto. Acreditamos que se esta cadeira se adaptasse àquelas lecionadas no 1º ano, nomeadamente AOCO e MPCP, todos saíram beneficiados, uma vez que os alunos estariam, provavelmente, mais à vontade com a transição. Sem dúvida que a transição entre as unidades curriculares é bastante brusca.

Em suma, LCOM foi uma cadeira desafiante – exigindo bastante tempo e esforço para que tudo fosse cumprido, mas ao mesmo tempo gratificante pelas aprendizagens adquiridas ao longo destes meses.

Apêndice

Flood fill Algorithm – how to implement fill() in paint? 19 de Novembro de 2020. https://www.geeksforgeeks.org/flood-fill-algorithm-implement-fill-paint/ (acedido em 1 de Janeiro de 2021).