Connect 4

LCOM – Projeto Final

Turma 11

Grupo 1

Realizado por:

Fernando Rego - up201905951@edu.fe.up.pt

Gonçalo Ferreira - up202004761@edu.fe.up.pt

João Duarte - up201707984@edu.fc.up.pt

Pedro Macedo - up202007531@edu.fe.up.pt

Índice

Introdução

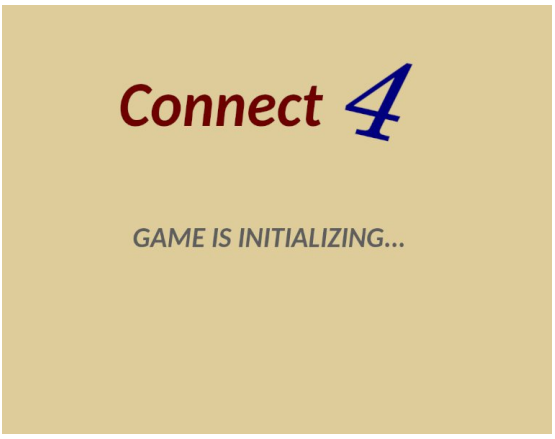
Para o nosso projeto final, decidimos criar um jogo em 2D, tradicionalmente conhecido como 4 em linha, ou “connect 4” em inglês. Neste jogo de dois jogadores, em que cada um tem uma cor diferente associada, (no nosso caso utilizamos azul e vermelho).

O principal objetivo é, de forma alternada, preencher uma posição da grelha de jogo com uma peça semelhante a uma moeda.

O primeiro jogador a obter 4 peças da mesma cor em posições contiguas ganha o jogo. Haverá empate caso não existam mais posições disponíveis a preencher na grelha e nenhum dos jogadores tenha conseguido preencher as 4 posições contiguas da sua cor.

Instruções de Utilização

1. Inicialização do Programa

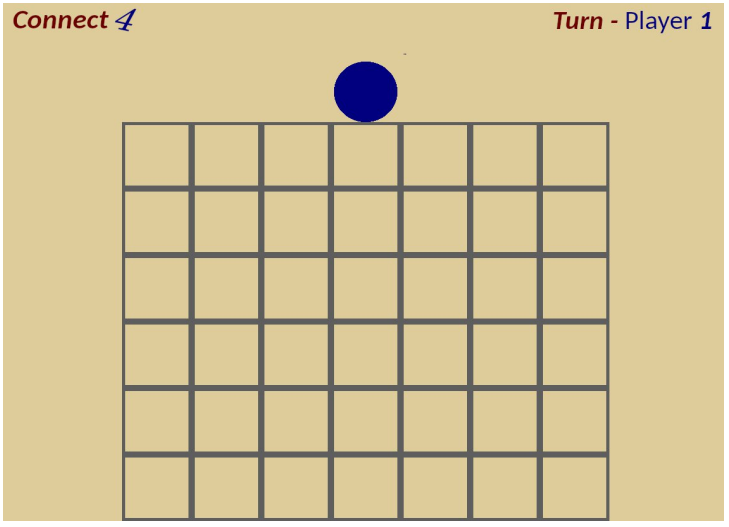


Ao iniciar o jogo, é mostrado inicialmente o menu inicial, onde é possível jogar (clicando em Play), ou ir para a secção Rules (clicando em Rules), para obter informação detalhada sobre as regras e os controlos, ou sair do jogo (clicando em Quit). Para navegar neste menu deve ser utilizado os movimentos do rato para mover o cursor, e o clique no botão esquerdo para selecionar a opção desejada.

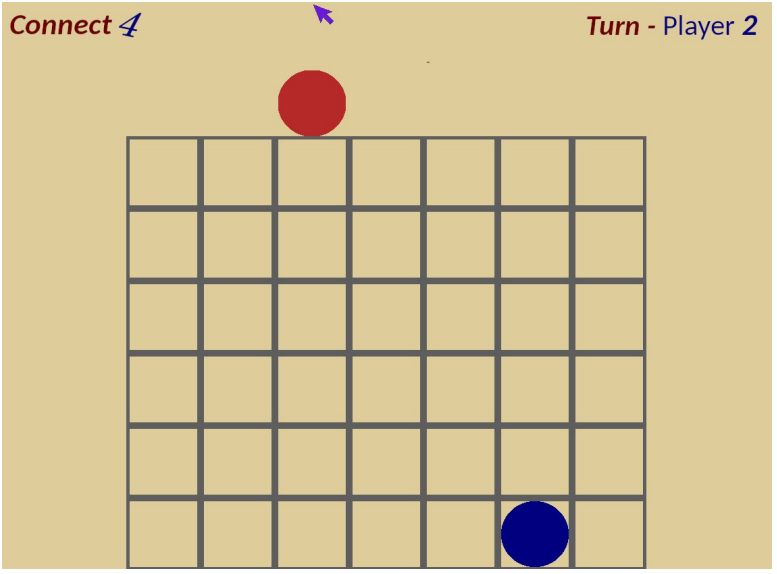
2. Menu Inicial

Ao iniciar o jogo, é mostrado inicialmente o menu inicial, onde é possível jogar (clicando em Play), ou ir para a secção Rules (clicando em Rules), para obter informação detalhada sobre as regras e os controlos, ou sair do jogo (clicando em Quit). Para navegar neste menu deve ser utilizado os movimentos do rato para mover o cursor, e o clique no botão esquerdo para selecionar a opção desejada.

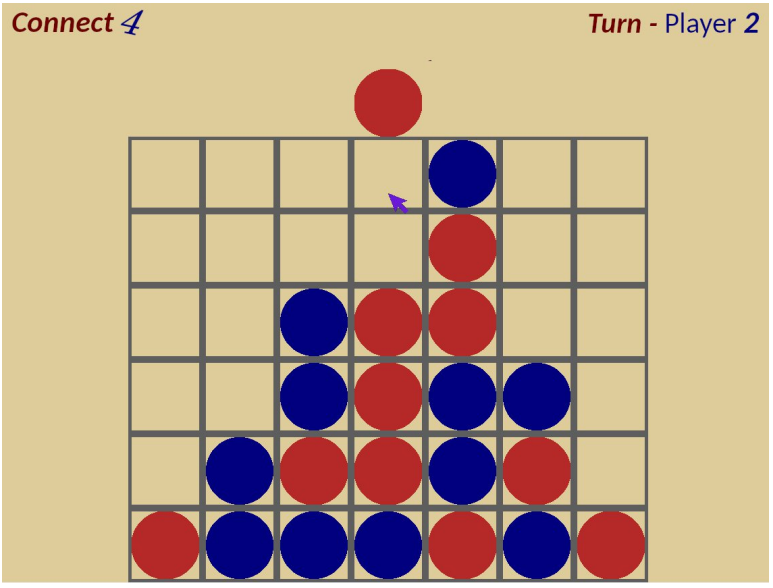
3. Play



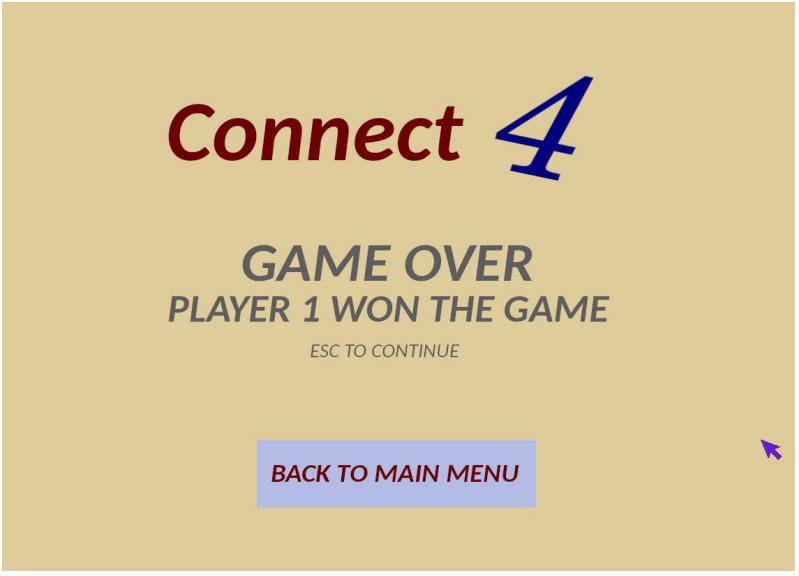
Quando se seleciona a opção Play, iniciamos o jogo com o tabuleiro vazio, na vez do Player 1 jogar. Utilizando os controlos definidos no menu Rules e selecionando a coluna desejada, após pressionar a barra de espaços a nossa peça será colocada na coluna escolhida.



Após a jogada do jogador 1 (azul), entra a vez do jogador 2 (vermelho) que fará o mesmo processo mas através da utilização do rato, ao contrário do jogador 1 que utiliza apenas o teclado.



Após a jogada do jogador 2, a vez retorna novamente ao jogador 1, e o jogo continua de forma fluente até existir um vencedor, ou um empate.



No caso de um dos jogadores conseguirem preencher 4 posições contiguas (verticais, horizontais ou diagonais), o jogo terminará demonstrando o jogador que venceu, podendo pressionar a tecla ESC para voltar ao menu principal, ou utilizar o rato para selecionar a opção “Back to main menu”.

4. Rules Menu

Quando se seleciona a opção Rules, é carregado um menu com uma pequena introdução e descrição do jogo.

Aqui é possível verificar quais controlos são necessários para ambos os jogadores disfrutarem do jogo. O Player 1 utilizará o teclado através das teclas “arrow” esquerda, ou direita de forma a selecionar a coluna que pretende jogar, e através da barra de espaços para largar a peça na coluna desejada.

Quanto ao player 2, este utilizará o rato, onde poderá simplesmente colocar o cursor na coluna desejada e um simples click largará a peça na coluna desejada.

A opção de pausar o jogo está disponivel através da tecla ESC (escape).

Para retornar ao menu inicial basta com o rato selecionar a opção “Back to Main Menu”.

Estado do Projeto

DISPOSITIVO	FUNCIONALIDADE	INTERRUPÇÕES
Timer	Controla o frame rate do jogo e as animações do movimento das peças	Sim
Keyboard	Selecionar a coluna em cada movimento do jogo, e alguns controlos no menu.	Sim
Mouse	Interação em cada menu para selecionar uma opção, e utilização pelo jogador 2 na seleção das colunas.	Sim
Graphics Card	Apresenta a interface do jogo.	Não
Real Time Clock	Apresentar o dia e a hora nos menus.	Sim

1. Funcionalidades

1.1 Menus

Foram desenhados e implementados 4 menus para fases diferentes do programa. Um menu principal, um menu para a apresentação do jogo e das suas regras, um menu de pausa e um menu onde está apresentado o resultado final de cada jogo

Cada menu contém a indicação da hora e data atual obtida pelo device RTC (detalhado mais à frente no relatório)

1.2 Jogo

O jogo implementado conta com um tabuleiro onde irão ser colocadas as peças em cada movimento do jogador. Cada jogador utiliza um device diferente para controlar e realizar a jogada, sendo que o primeiro jogador utiliza o keyboard (setas para a esquerda e direita para escolher a coluna e espaço para realizar jogada) e o segundo jogador utiliza o mouse (posição do rato para escolher a coluna e left-click para realizar jogada)

1.3 Animação

Para cada jogada feita pelos jogadores, uma animação simples é feita, utilizando o timer, para dar o efeito da queda vertical de cada peça em cada coluna do tabuleiro

1.4 Funcionalidades Adicionais

No desenvolvimento do projeto deparamo-nos com um problema comum devido ao desenho de muitos pixeis:o flickering. Assim, para tentar reduzir ao máximo foi implementado o **triple buffering** (com double buffering por vezes ainda se notava este problema) através de page flipping, ou seja, alternar entre os 3 buffers alocados inicialmente, sendo que apenas é mostrado/alternado após o buffer ser completamente desenhado. Após esta implementação detetamos um melhoramento substancial a nível gráfico do programa.

1.5 Funcionalidades por Implementar

Devido a restrições de tempo não foi possível a implementação do device **Serial Port.** Este device podia ter sido desenvolvido com o intito de jogar entre dois PCs diferentes, ou neste caso, entre duas Virutal Machines ligadas pela serial port.

2. Timer

O timer é usado para controlar o frame-rate, dando um valor fixo à frequência do timer para consistentemente gerar interrupções a um número fixo por segundo, no nosso caso 60. Estas interrupções são essencialmente utilizadas para atualizar o ecrã. Outra funcionalidade conseguida com o timer foi também demonstrar uma animação de jogo a cada jogada para obter o efeito de a peça estar a “cair” na posição desejada do tabuleiro.

A implementação do device timer utilizada no projeto foi muito semelhante à implementação utilizada no lab2.

3. Keyboard

O keyboard é usado para movimentar a peça do jogador 1, usando as teclas das setas, e a barra de espaços para largar a peça. Outras funcionalidades conseguidas através do keyboard é usando a tecla ESC, pois tanto volta para o estado anterior no caso de estarmos em um menu como também para abrir o menu de pausa caso seja necessário parar o jogo.

Todas as funcionalidades do keyboard foram conseguidas através do uso de interrupções geradas através dos *clicks* nas teclas e dando handle a cada interrupt para realizar a ação necessária. A implementação deste foi muito semelhante ao lab3 onde a alteração mais significativa foi a remoção das funções onde, em vez de interrupções, era utilizado polling.

4. Mouse

O rato é utilizado tanto para a navegação do utilizador pelos menus do jogo como para o jogador 2 realizar a sua jogada. Nos menus de jogo o rato é utilizado para dar *clicks* nos botões para alterar o programa de estado. Uma funcionalidade adicional implementada foi o *hover* nos botões quando a posição do rato coincide com a área ocupada pelos botões sendo que a cor destes altera de forma a fornecer um efeito visual positivo para o utilizador. Por outro lado, em cada jogada feita pelo jogador 2, a posição do rato é essencial para escolher a coluna pois, no momento em que o utilizador faz *left-click*, a coluna do tabuleiro é calculada em função da posição do mouse.

Como no caso do teclado, a implementação do rato foi muito semelhante à implementação do lab4, onde foi feita a implementação de uma função para dar *enable/disable* ao *data reporting* e para cada *mouse packet* gerado (obtido através de handle de cada interrupção gerada pelo dispositivo), as coordenadas do rato são atualizadas de acordo com o movimento do device.

5. Graphics Card

O dispositivo mais importante na realização deste projeto é sem dúvida a placa gráfica. Este dispositivo fornece ao utilizador uma experiência visual através do desenho de sprites e do uso de uma fonte para desenhar caracteres (utilizados para desenhar a data atual), o que seria impossível sem implementação da placa gráfica.

O modo de video programado para a utilização da placa gráfica é o modo 0x14C que fornece uma resolução de 1152x864 em que cada cor é representada por 32 bits da forma

(8:)8:8:8. Para além disso foi implementado o **tripple buffering** para obter uma melhor sensação visual para o utilizador, ultrapassando o problema de flickering.

A implementação deste device para o projeto foi feita com base no que foi desenvolvido para o lab, com a adição de page flipping e de algumas otimizações em relação à forma que se desenha cada sprite.

6. RTC

O dispositivo RTC, apesar de não ter sido abordado nos labs, é utilizado para mostrar a data atual em cada menu presente no jogo.

A implementação do dispositivo, para a finalidade pretendida, foi relativamente simples sendo que apenas existe a subscrição de interrupções de updates periódicos, o suficiente para conseguirmos a obtenção da data atual do sistema.

Estrutura e Organização do Código

1. Proj

Peso Relativo: 5%; Implementado por: Fernando Rego

Módulo principal que contém o ciclo principal do programa. Este módulo contém as seguintes funcionalidades:

- Inicializar a placa gráfica no modo pretendido
- Desenhar um menu de inicialização do jogo
- Subscrever interrupções de todos os dispositivos usados, exceto placa gráfica
- Inicializar todos os recursos utilizados no programa como menus, o jogo e os sprites que cada um contém
- Destruir todos os recursos quando o programa termina a execução
- Remover todas as subscrições feitas na terminação do programa
- Retornar ao modo de texto quando o programa termina a execução

O ciclo principal é onde chamadas ao driver_receive() são feitas, que nos fornece informação do evento ocorrido para, através de chamadas de funções de handle presentes no módulo Events (posteriormente explicado) alterar o estado do programa.

2. Lib

2.1 Devices

Peso Relativo: 25%; Implementado por: Todos os membros

Módulo onde está presente a implementação dos dispositivos Timer, Keyboard, Mouse, Graphics Card e RTC abordados previamente.

2.2 Sprite

Peso Relativo: 5%; Implementado por: Fernando Rego

Módulo onde está presente a struct *sprite* que guarda a seguinte informação:

- (uint8_t *) mapa da sprite que é loaded usando a função xpm_load()
- (xpm_image_t) imagem da sprite que guarda informação sobre a sprite
- posição do pixel (x,y) do canto superior esquerdo da sprite

Para além da struct fornece métodos para criar uma nova sprite, destruir o sprite, e alterar a posição do canto superior esquerdo da sprite de duas formas diferentes, a primeira é passando como argumento uma posição (x,y) completamente nova e a segunda é passando como argumento o deslocamento (delta_x, delta_y) onde à posição da sprite é adicionado este deslocamento.

2.3 Utils

Peso Relativo: 2%; Implementado por: Todos os membros

Módulo onde estão presentes as funções auxiliares implementadas para a realização dos labs como também uma função auxiliar onde, a partir de duas sprites, sprite do mouse e sprite de um botão, verifica se a posição do mouse coincide com a área ocupada pelo botão retornando *true* se isso se verificar e *falso* caso contrário

3. Events

Peso Relativo: 15%; Implementado por: Fernando Rego

O módulo Events é sem dúvida o mais importante do programa inteiro pois está encarregado de dar handle a todo o tipo de eventos necessários no decorrer do programa. Assim este módulo tem uma função para cada par estado de jogo – dispositivo, ou seja, se o jogo se encontrar no estado PAUSE_MENU e o mouse gerar uma interrupção, então a função associada a esse estado e dispositivo será chamada para dar handle do evento. Cada função retorna um novo estado de jogo atualizado.

Adicionalmente é neste módulo que estão definidos no *enum* GAME_STATE todos os estados do programa essenciais para a máquina de estados. Os possíveis estados são:

- IN_GAME – está a ocorrer um jogo
- END_GAME – terminou o jogo e é apresentado o resultado
- ANIMATION_GAME – está a ocorrer uma animação
- MAIN_MENU – apresentado o menu principal
- PAUSE_MENU – apresentado o menu de pause
- RULES_MENU – apresentado o menu de regras
- QUIT – indicação para terminar execução do programa

4. Game

4.1 Game

Peso Relativo: 20%; Implementado por: Fernando Rego

Módulo fundamental do jogo que guarda todas as informações necessárias sobre do estado de cada jogo através da struct *game* e guarda todas as sprites associadas ao jogo para desenhar quando e onde necessário. Contém também uma struct *anim* utilizada para a animação das peças a cada jogada e, por fim, toda a lógica do jogo necessária para o jogo connect 4.

A struct *game* contém um array 2D que simboliza a board com 6 linhas e 7 colunas onde cada elemento do array apenas pode conter os valores 0 (lugar vazio), 1 (ocupado peça do jogador 1) ou 2 (ocupado por peça do jogador 2). Contém também um inteiro que representa a vez do jogador podendo então ser 1 (vez do jogador 1) ou 2 (vez do jogador 2) e um outro inteiro que representa a coluna selecionada na jogada atual. Por fim contém um pointer *display_buffer* que é utilizado para otimizar o desenho do jogo no menu. Esta otimização é explicada detalhadamente na secção de Detalhes de Implementação.

A struct *anim* contém a sprite que realiza a animação, a posição inicial (x,y) e a altura final da peça. Sendo que a animação consiste na queda vertical da peça numa coluna do tabuleiro, nada mais é preciso para realizar a animação pois só é necessário verificar a posição associada ao eixo vertical.

4.2 Game Utils

Peso Relativo: 3%; Implementado por: Fernando Rego

Módulo que fornece funções auxiliares para a realização do jogo, mais concretamente duas funções para obter a posição no ecrã do canto superior esquerdo (x,y) de cada casa do tabuleiro de jogo e consequentemente desenhar a sprite pretendida no sítio correto.

4.3 Game End Menu

Peso Relativo: 5%; Implementado por: Pedro Macedo, Gonçalo Ferreira e João Duarte

Módulo semelhante ao Menu mas que fornece a informação sobre o final de cada jogo. Como o módulo Menu, guarda as sprites necessárias para apresentar o resultado final do jogo e adicionalmente um inteiro que pode ter os valores 1 (jogador 1 ganhou o jogo), 2 (jogador 2 ganhou o jogo) ou 3 (jogo acabou em empate) para decidir qual sprite desenhar no ecrã. Por fim guarda a sprite que representa o mouse para facilitar a navegação do utilizador pelo menu.

5. Menu

5.1 Font

Peso Relativo: 5%; Implementado por: Todos os membros

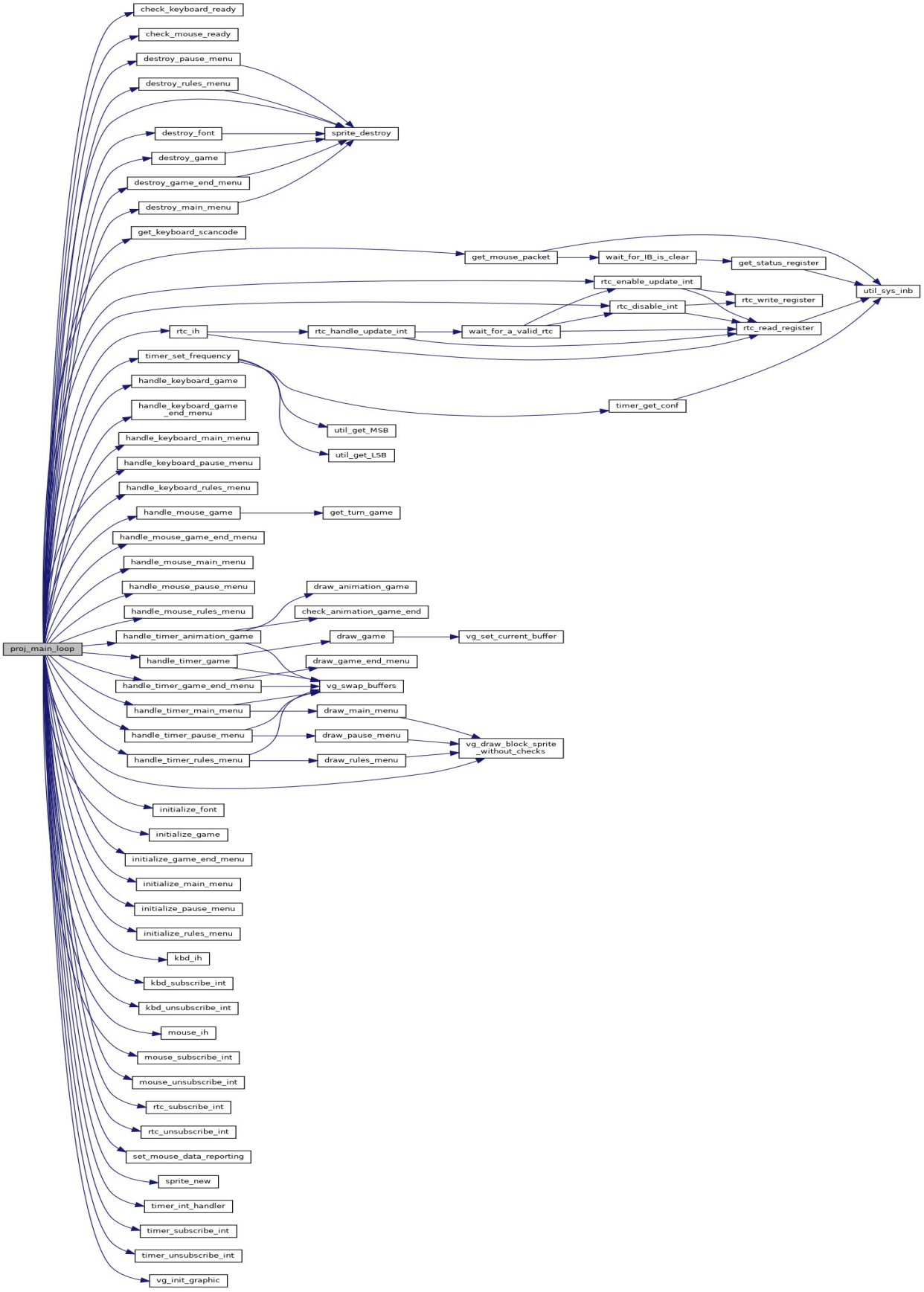
Módulo que contém sprites que representam todos os caracteres necessários para representar a data atual, e de funções utilizadas para desenhar a data numa posição fixa em cada menu.

5.2 Menus

Peso Relativo: 15%; Implementado por: Pedro Macedo, Gonçalo Ferreira e João Duarte

Módulo com todos os menus principais utilizados no jogo, o menu principal, menu de regras e o menu de pausa. Cada um destes menus contém as diferentes sprites para desenhar o menu, como por exemplo o background e os botões de cada um. Por fim guarda também a sprite que representa o mouse para facilitar a navegação do utilizador pelos menus.

6. Gráfico de Chamadas de Funções



Detalhes de Implementação

1. Máquina de Estado

Como foi referido anteriormente, o nosso projeto conta com uma máquina de estados com o objetivo de guardar o estado atual de jogo e, consoante o estado, ser apresentado ao utilizador tudo o que está relacionado com o mesmo estado.

2. Programação Orientada a Objetos

No desenvolvimento deste projeto, foi utilizado formas semelhantes ao que acontece a nível de programação orientada a objetos, sendo que cada módulo pode ser visto como uma classe em que os elementos estáticos podem ser vistos como *fields* da classe e podem ser obtidos ou modificados através de getters e setters. As próprias structs também podem ser vistas como classes que apenas guardam informações relativas ao próprio objeto ou struct.

3. Otimização no Desenho do Jogo

Anteriormente foi referido que a struct game continha um pointer *display_buffer*. Este buffer existe apenas para conseguir-mos uma otimização no desenho do jogo. Sendo que no jogo Connect 4, cada vez que uma peça é colocada no tabuleiro, ela permanece sempre na mesma posição do tabuleiro, pensamos numa possível otimização. Assim este display_buffer tem o tamanho de qualquer um dos três buffers utilizados pela placa gráfica, ou seja o tamanho suficiente para desenhar todo o ecrã. No início de cada jogo é desenhado o background e o tabuleiro vazio no buffer, e a cada jogada feita ao longo do jogo é desenhada a nova peça na posição correta no mesmo buffer. Assim, existe uma diminuição muito grande na quantidade de sprites desenhadas na memória gráfica pois, em vez de por cada peça no tabuleiro ser desenhado uma sprite, é apenas copiado diretamente o conteúdo do *display_buffer* para a memória utilizando a função *memcpy()*.

Conclusões

Após o desenvolvimento de todo este projeto podemos afirmar que a cadeira é sem dúvida das mais interessantes que tivemos e a que mais nos desafia.

O nível deste projeto podia ser ainda mais elevado com uma futura implementação do dispositivo serial port, pois seria possível jogar entre duas Virtual Machines o que atualmente no mundo real é um requisito praticamente obrigatório para jogos com vários jogadores.

A programação de dispositivos, numa primeira fase não é trivial devido à sua complexidade, porém a partir do momento em que vamos praticando e resolvendo os “labs”, acaba por ser tornar cativante e interessante. A programação de baixo nível permite-nos perceber de forma muito mais clara e objetiva tudo o que acontece em todo o nosso programa e fornece uma ideia muito mais pormenorizada de todo o processo desenvolvido. Adicionalmente, com o que cada dispositivo permite fazer, obtemos um leque enorme tanto de funcionalidades que podemos implementar como de possibilidades de resolução de problemas, embora umas estejam mais corretas do que outras.

Porém, em alguns casos, o facto de por vezes torna-se complicado entender totalmente os guiões transforma o desafiante em frustrante, este será um ponto negativo da cadeira o que dificulta bastante a resolução dos labs.

Em suma, podemos afirmar que de facto é uma unidade curricular que pede bastante tempo de prática resultando numa carga horária grande, pelo que acaba por ser difícil conciliar com as restantes cadeiras.