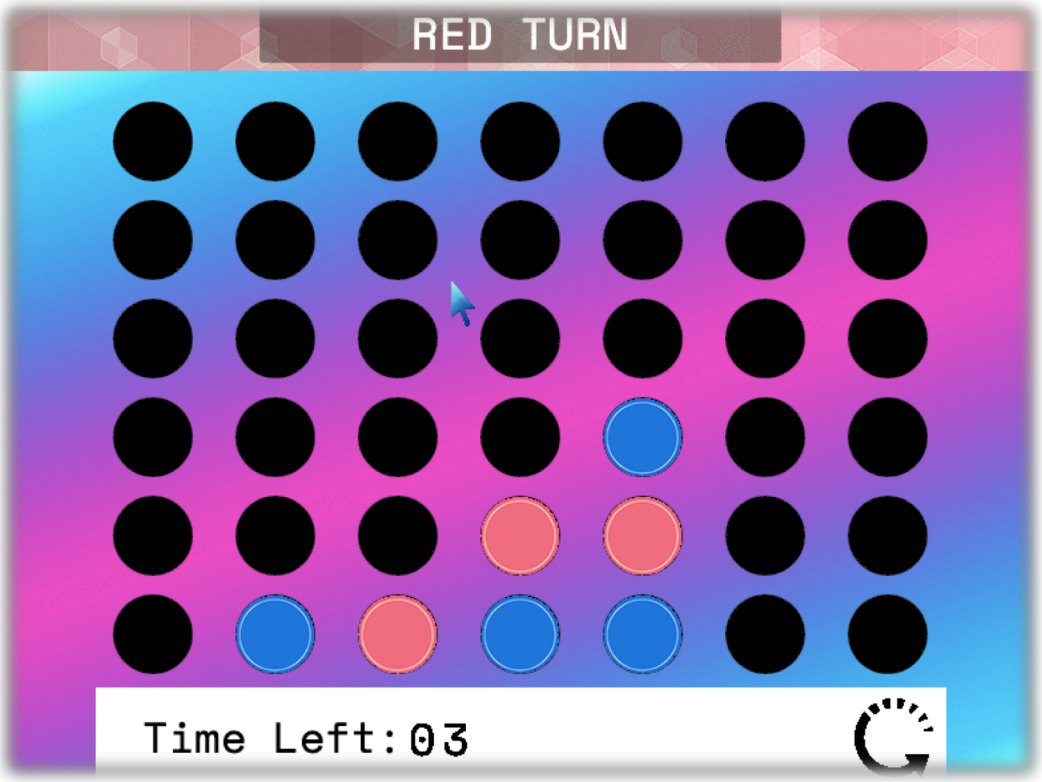
Embernite



Laboratório de Computadores 19/20

Turma 5 – Grupo 12

06 de Janeiro de 2019

Daniel da Silva Gonçalves up201809384@fe.up.pt
Guilherme Candicella Calassi up201800157@fe.up.pt

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

O presente documento servirá não só como um mero relatório a ser associado ao nosso projeto da cadeira de Laboratório de Computadores (19/20), como também servirá de componente medular de coadjuvação aos respetivos docentes. Acreditamos que um relatório tem o papel principal de não só 'reportar' os elementos essenciais e objetivos atingidos no objeto a que se enquadra, como também tem o dever de ajudar leitores e docentes a entender com maior facilidade as informações que se pretendem transmitir. Ou seja, que abuse de um ambiente introdutório.

Assim, e com o objetivo de exercer as utilidades supracitadas, este relatório será dividido de acordo com cinco secções dominantes:

1. **Temática**
2. **Background da Temática**
3. **Máquinas de Estados**
4. **Módulos e Ficheiros**
5. **Estruturação do Código**

Na primeira secção, procuraremos colocar na mesa as regras do jogo, os seus comandos assim como funcionalidades que irão reger a forma como o jogo se desenrolará. Na secção 2 e 3, iremos abordar (de forma superficial) a forma como o nosso código consegue realizar muitas das funcionalidades abordadas. Apenas será fornecida uma base teórica de aplicação, e nenhuma citação de código será levantada para o efeito. No entanto, e para facilitar o trabalho dos nossos docentes (e evitar confusão), citaremos alguns dos títulos das funções que chamamos para realizar algumas operações, ficando a forma como é empregue no ficheiro correspondente.

Relativamente à quarta função, que interliga com a quinta, serão apresentados cada módulo do nosso código assim como os ficheiros ligados a eles. A secção de estruturação de código dará luz a um diagrama que apresentará todas as ligações entre cada função. Abordará, ainda, aspetos essenciais.

Espera-se que no final as dificuldades em entender o nosso código sejam relativamente menores com o auxílio a este documento. Sublinha-se ainda a necessidade absoluta em ler a quinta secção devido ao uso desmoderado de variáveis globais no nosso projeto (isto evitou manipulação de apontadores e facilitou bastante o arranjo de conexões entre funções, mas admitimos lamentavelmente que afetou a clareza do código nas primeiras linhas). No entanto, é garantido que após entender o significado e uso de cada variável global, seja fácil entender o resto do projeto.

2 Temática

O projeto que é alvo de abordagem deste relatório não é nada mais, nada menos, que uma aplicação do jogo *4 in a row*, também comummente designado de '4 em linha'[1] em linguagem Portuguesa.

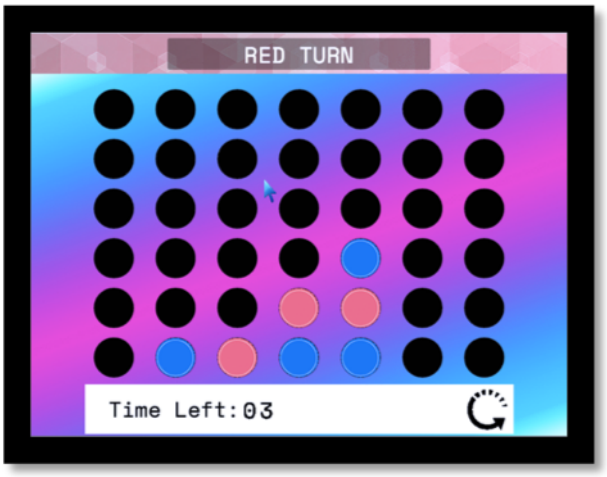


Figura 1: Interface do jogo

Tal como apresentado ao lado (e presente em maior dimensão na capa deste documento), o jogo é composto por um tabuleiro com buracos dispostos numa matriz 6x7 igualmente espaçada tanto na vertical como na horizontal. É também composto por dois tipos de fichas: azuis e vermelhas que são atribuídas aos jogadores com a respetiva cor.

Para jogar, tudo o que se tem de fazer é largar uma ficha no topo do tabuleiro. Esta cairá até atingir o fundo da coluna ou colidir com uma ficha qualquer já presente na coluna, ficando em cima desta. Se a coluna já estiver preenchida, não deverão ser adicionadas quaisquer fichas. Não há qualquer outra maneira de colocar a ficha a não ser pela forma supracitada. Em suma, podemos concluir que todas as colunas terão de ser preenchidas de baixo para cima e nunca o contrário. Cada jogada de cada jogador é alternada e nunca poderão ser largadas duas fichas em simultâneo.

O jogo pode terminar com um vencedor ou num empate. O critério de vitória é dado pela existência de quatro fichas consecutivas da mesma cor na mesma linha, coluna ou diagonal. O vencedor será aquele correspondente à cor dessas mesmas fichas. Um empate será dado caso toda a tabela esteja preenchida e não se consiga verificar o critério de vitória (para ambos os jogadores).

A funcionalidade de restart existe no jogo físico e é dada pela abertura de uma tampa por debaixo do tabuleiro que faz todas as fichas cair. Decidimos implementá-la para este projeto (mas sem a animação de queda das fichas), limpando todas as fichas do tabuleiro e recolocando o jogador pré-definido inicial.

Espera-se que no final as dificuldades em entender o nosso código sejam relativamente menores com o auxílio a este documento. Sublinha-se ainda a necessidade absoluta em ler a quinta secção devido ao uso de algumas variáveis globais no nosso projeto (isto evitou manipulação de apontadores e facilitou bastante o arranjo de conexões entre algumas funções). Todavia, é garantido que após entender o significado e uso de cada variável global, seja fácil entender o resto do projeto, pelo que será dada uma explicação singular e resumida de cada uma.

[1] No jogo original não é habitual haver um limite de tempo para cada jogada, no entanto decidimos estender a utilidade do timer através da implementação de um. Adicionalmente, por definição, um dos jogadores é sempre o inicial. Tomou-se o azul por convenção.

3 Background da Temática

3.1 Jogo

Tendo em mente as regras do jogo, podemos agora explicar a forma como estas são aplicadas assim como a elaboração do nosso jogo e código. Note-se que apesar de fazermos uso de expressões como 'ambos os jogadores' e 'cada jogador', este projeto não usufrui das funcionalidades do Serial Port dado que fugiria da natureza do jogo *4 in a row* que usualmente é feito com dois jogadores a fazerem uso do mesmo equipamento.

Primeiramente, fizemos uso de um cursor *(definido em sources/cursor.xpm)* que nos levou a aproveitar o uso do mouse. Por definição, o cursor começa sempre nas coordenadas (0,61) e é impossibilitado de entrar na caixa que disponibiliza a cor do turno. Um estilo de cursor diferente é visualizado sempre que o cursor se encontra em cima de uma coluna. Tem a aparência de uma seta a apontar para baixo *(definido em sources/down_arrow.xpm)* tal que, caso seja pressionado o botão esquerdo do rato (LMB), simulará a ação de largar uma ficha em cima da respetiva coluna.

O cursor ajuda o jogador a guiar-se pelo jogo; seja para selecionar uma coluna ou para carregar no botão 'restart' que se encontra no canto inferior direito do ecrã. Também poderá ser feito uso do keyboard caso não haja qualquer preferência pelo rato por parte do jogador.

O uso de teclas de movimento (A ou D, ou ←, ou →) restringem a posição do cursor para o centro [horizontal] do topo de cada coluna. Se o utilizador pressionar uma das teclas de movimento, o cursor mexe-se para a coluna adjacente. Teclas adicionais foram adicionadas para incrementar o número de funcionalidades e, na totalidade, durante o jogo faz-se uso das seguintes teclas:

- **[M][2]:** A (ou ←) – move o cursor para a coluna adjacente esquerda.
- **[M]:** D (ou →) – move o cursor para a coluna adjacente direita.
- **[M]:** Enter (ou spacebar) – faz cair uma ficha da mesma cor do turno na respetiva coluna.
- **[M]:** R – Executa um reset ao jogo.
- **[M]:** Q – Sai do jogo e encaminha ambos os jogadores para o menu.
- **[B][3]:** ESC – Termina o programa (de forma ordeira).

Ambas as funções de movimento do cursor e reação do teclado são efetuadas pela função: 'game_react()' *(game.c/.h)*. Existem também funções adicionais, especialmente a função 'delete_cursor()' *(cursor.c/.h)* que limpa a zona em arredor do cursor antes de mover o cursor para a próxima posição. Se for pressionada a tecla ENTER ou BARRA DE ESPAÇO (enquanto estiver a trabalhar no teclado) ou for pressionado LMB com o cursor em cima de uma coluna, é chamada a função 'process_fallingChip()' que processa a animação da queda da ficha no tabuleiro a um frame rate de 60 fps (macro).

Como já mencionado anteriormente, entre cada turno existe um tempo limite para as jogadas. Este é definido pela macro no ficheiro *'proj_essentials.h'*: DEFAULT_SECONDS. O valor definido por esta corresponderá ao limite de tempo de cada turno e não poderá ser inferior a 2 para garantir o normal funcionamento do jogo.

[2] MakeCode
[3] BreakCode

A cada 60 ticks, correspondente à diminuição do tempo restante por 1 segundo, o programa limpa o cursor e os números atuais antes de reimprimir os próximos números representantes do novo tempo restante. A manipulação do tempo que resta é dado pela variável global 'remainingSeconds'. As funções que tratam deste sistema é dado por:

- **timer_int_handler()** – timer.c/.h (Lab 2)
- **deleteCursor()** – cursor.c/.h
- **project_cursor()** – proj_essentials.c/.h
- **remaining_time()** – graphic_management.c/.h

No final do jogo é sempre apresentada apenas uma de duas imagens[4] (exceto se for um empate, para esse caso decidiu-se que o header mudaria temporariamente para 'DRAW' durante 3 segundos antes de realizar um reset ao jogo.). Cada imagem apresenta o vencedor e tem duas opções de saída ou de reset. O processamento destas imagens é chamada pela função 'end_screen()' no(s) ficheiro(s) *game.c/.h*.

[4]Antes de aparecer essas imagens e, caso seja detetada vitória, serão apresentados quatro círculos dando destaque à sequência que marca as 4 fichas que decidiram a vitória. Este processamento é dado pela função: victory_process() presente no ficheiro *proj_essentials.c/.h*

3.2 Menu

Figura 2: Menu Inicial

Para melhorar a aparência do jogo, foi adicionado o MENU onde aproveitamos para implementar a apresentação da data atual e fazer uso do Real-Time Clock (RTC). O menu é sempre a apresentação inicial do programa.

Através da opção *PLAY* podemos iniciar o jogo. Através da opção *EXIT* podemos terminar o programa.

Semelhantemente ao jogo, o menu pode ser controlado pelo keyboard, **mas não pode ser controlado pelo mouse**. As teclas que podem ser utilizadas no menu são:

- **[M][5]:** W – move para a opção acima (ou fundo, se já estiver no topo).
- **[M]:** S – move para a opção abaixo (ou topo, se já estiver no fundo).
- **[M]:** Enter (↩) – seleciona a opção.
- **[B][6]:** ESC – termina o programa.

O RTC, por sua vez, é apresentado no canto inferior esquerdo e é atualizado sempre que envia uma interrupção de término de atualização. Todas as funções de manuseamento do RTC estão disponíveis em *rtc.c/.h*. A mesma semelhança é aplicável ao Keyboard, Mouse e Timer, que também funcionam todos com base em interrupções.

Em função destes dados é possível atribuir a utilidade a cada dispositivo de I/O assim como indicar se este usufrui dos interrupts ou não.

Tabela 1: Informações Dispositivos

Dispositivo	Utilidade	Interrupts
Timer	Tempo limite de cada jogada e frames durante a queda das fichas no tabuleiro ← process_fallingChips()	V
Keyboard	Selecionamento de Colunas e/ou Opções	V
Mouse	Movimento do cursor e uso dos botões para selecionar as colunas.	V
RTC	Disponibilização da hora e data atual. Atualiza com interrupções.	V
VBE	Apresentação do MENU, JOGO e suas funcionalidades.	F

A memória virtual é sempre atualizada com **double buffering**. Fizemos também uso do modo gráfico 0x115 que trabalha com 3 bytes por pixel e contém uma resolução acessível de 800 por 600.

[5] MakeCode
[6] BreakCode

4 Máquinas de Estados

Para nos ajudar, o programa faz uso constante de duas máquinas de estado regidas por enumerações. Todas estas se encontram definidas no ficheiro: proj_essentials.h e globalmente definidas no respetivo ficheiro.c.

- A primeira enumeração, designada de 'program_state' lida simplesmente com o estado do programa em geral. Dada a sua simplicidade, a máquina apenas indica se estamos no MENU ou dentro do JOGO.

  O diagrama de estados abaixo resume de forma breve a simulação desta máquina de estados:

  Est. 1 = CURRENTLY_IN_MENU Est. 2 = CURRENTLY_IN_GAME

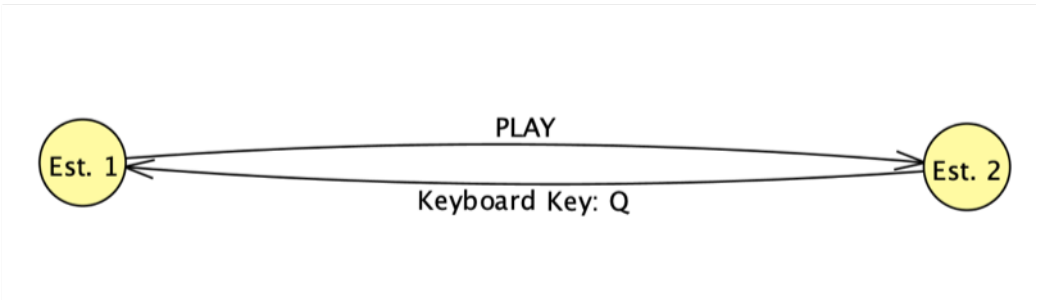


Figura 3: Máquina de Estados Geral

- A segunda enumeração designada de 'game_state' lida com o jogo na sua totalidade. De facto, todo o jogo é controlado por esta máquina de estados e é também esta que estabelece se o jogo terminou, ou não.

  O diagrama de estados (incompleto) abaixo resume esta máquina:

  Est. 1 = BLUE_PLAYING Est. 2 = RED_PLAYING Est. 3 = BLUE_WON

  Est. 4 = RED_WON Est. 5 = PUTTING_CHIP Est. 6 = REVEALING_PATH

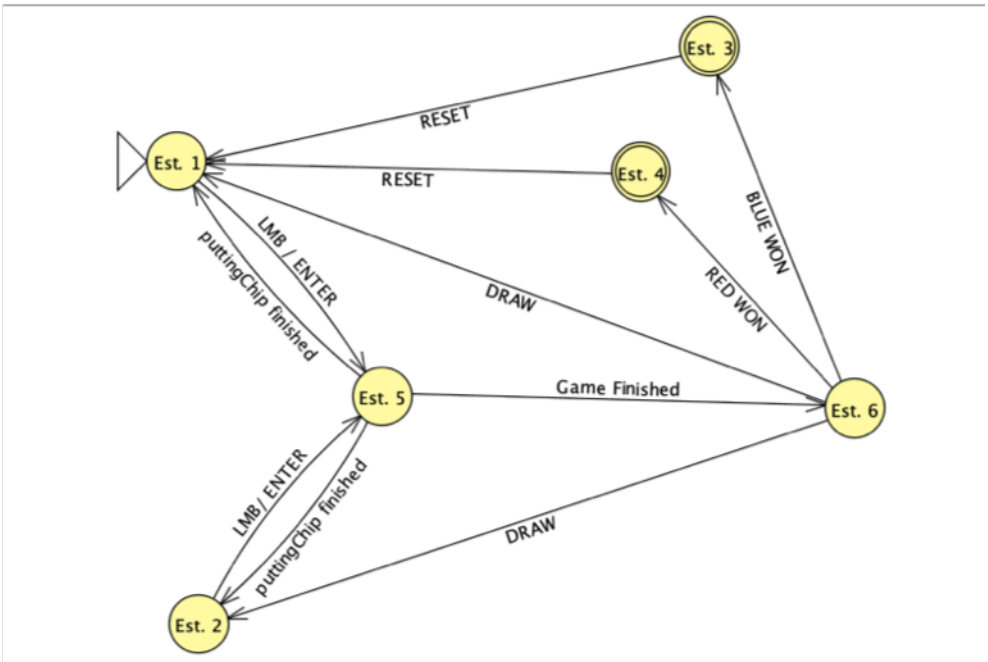


Figura 4: Máquina de Estados Reguladora

5 Módulos e Ficheiros

Relativamente ao projeto em causa, existem uma variedade de módulos e ficcheiros que compõem a estrutura deste. Nesta secção, e tal como o título sugere, trataremos de atribuir breves descrições das funções e utilidades de cada módulo.

Apesar de brevidade da explicação dada neste documento para cada módulo, uma informação mais detalhada pode ser encontrada dentro de cada ficheiro.h.

Project Essentials – O seguinte módulo cujos ficheiros são: proj_essentials.c/.h, tem como puro intuito conter a função essencial que contém o loop de interrupções e macros. Adicionalmente este ficheiro contém [quase] todas as variáveis globais. Alternativamente, este ficheiro pode ser visto como o centro do programa, responsável por todo o controlo em função de cada interrupção que recebe.
Este módulo faz uso das seguintes funções:

- **floor()** - Calcula o valor 'floor' de um dado número.
- **ticks_fr()** – Calcula o número de ticks necessários para uma dada frame rate.
- **project_core()** – Esta função reina o programa, contendo o ciclo de interrupções e orientando todo o programa para as respetivas funções.
- **victory_process()** – Desenha anéis amarelos em torno das fichas que verificaram o critério de vitória.

RTC – O seguinte módulo cujos ficheiros são rtc.c/.h controla o Real-Time Clock (RTC). É também o ficheiro responsável por ativar e desativar as interrupções do RTC. São incluídas uma variedade de funções responsáveis pela leitura de data e tempo e escrita de comandos para os registos do RTC.
Este módulo faz uso das seguintes funções:

- **rtc_subscribe_int()** - Subscreve as interrupções do RTC.
- **rtc_unsubscribe_int()** – Cancela as subscrições das interrupções do RTC.
- **read_rtc()** – Lê um dado registo do RTC.
- **write_rtc()** – Escreve um valor de 8 bits para um dado registo do RTC.
- **isBCD()** – Analiza se os valores estão escritos em binário ou em BCD.
- **BCDtoBinary()** – Converte valores em BCD para binário.
- **getDate()** – Extrai a data do registo 'A' pertencente ao RTC.
- **getTime()** – Extrai o tempo do registo 'A' pertencente ao RTC.
- **setup_RTC()** – Prepara o RTC e habilita interrupções on update.
- **update_date()** – Atualiza a struct correspondente à data e tempo.

Game – O seguinte módulo cujos ficheiros são game.c/.h contém funções que lidam com a análise do estado do jogo. Exemplos são o processamento da queda das fichas, a reação do jogo perante certas interrupções e a ação de 'reset' do jogo.
Este módulo faz uso das seguintes funções:

- **game_react()** - Analiza e "chama" modificações gráficas com base na interrupção recebida e no estado atual do jogo.
- **process_fallingChip()** – Processa a animação da queda da ficha no tabuleiro.
- **draw_allChips()** – Mostra todas as chips largadas nas suas respetivas posições.
- **RESTART()** – Realiza um reset ao jogo, limpando as fichas.
- **checkWin()** – Verifica se um conjunto de quatro fichas cumprem o critério de vitória.

Graphic Management – O seguinte módulo cujos ficheiros são *graphic_management.c/.h* é o responsável por apresentar modificações gráficas com base nas mudanças que ocorrem no jogo. Este módulo encontra-se ligado diretamente com o módulo 'Game' (acima). Todas as suas funções encontram-se detalhadas no respetivo .h.

Cursor – Este módulo contém os ficheiros *cursor.c/.h* e são os responsáveis por conter as funções de manuseamento do cursor. Contém funções que detetam se o cursor se encontra em cima de dadas posições (e.g. útil para detetar se está dentro das colunas) e apaga o cursor de forma eficiente, limpando **apenas** uma pequena zona que se encontra em volta deste e voltando a colocar o que estava no seu background.

XPM – Faz uso dos ficheiros *xpm_management.c/.h* e são os ficheiros responsáveis por manusear os XPMs. O uso desta biblioteca é restrito meramente ao alocamento e libertação dos XPMs presentes dentro da pasta sources.

Para além destes módulos customizados foram incluídos os módulos utilizados nas aulas práticas anteriores, nomeadamente:

- (Lab 2) Timer → timer.c/.h
- (Lab 2) i8252
- (Lab 3) Keyboard Essentials → kbd_essentials.c/.h
- (Lab 4) Mouse → mouse_essentials.c/.h
- (Lab 5) Video Card (VBE) → videocard_essentials.c/.h

Estes últimos ficheiros não contêm qualquer elaboração se não aquela que foi submetida quando foi submetido o lab correspondente. Todas as funções que foram usadas nos labs estão também presentes nestes documentos e não foram modifica-das (exceto o draw_XPM() do *videocard_essentials.c* que necessitou de alguns ajustes).

6 Estruturação do Código

Foi mencionada na introdução a importância desta secção do relatório. Como desenvolvedor do código atônito deste projeto, devo salientar a necessidade da leitura dos header files para entender cada módulo dado que indicarão toda e cada funcionalidade o melhor que possível. No entanto, as variáveis globais (que tiveram como consequência uma enorme dificuldade em organizar o código) tornam-se bastante exageradas mas simplificaram enormemente a conexão entre funções. Para corrigir este defeito, criei a seguinte tabela que detalha todas as suas funcionalidades e tem como objetivo ajudar os docentes a guiar-se por estas variáveis caso falhe algum significado e/ou utilidade de alguma ao rever o código:

Variáveis globais compartilhadas (mais detalhado em *proj_essentials.c*):

Tabela 2: Tabela Variáveis Globais

Nome	Tipo	Utilidade
cur_xy[2]	int	Coordenadas atuais do cursor [x,y]
victorious_coordinates	int	Usado para memorizar os locais das fichas que determinaram a vitória
secondBuffer	uint8_t*	Apontador para o double_buffering Alocado dinamicamente
Game_background	uint8_t*	Apontador para um backup do background do jogo (tabuleiro)
Chips_background	uint8_t*	Apontador para um backup das fichas do jogo (e respetivas posições)
stateOfGame	enum	Máquina de estados do estado atual do jogo (game_state)
stateOfProgram	enum	Máquina de estados do estado atual do programa (program_state)
game_table[6][7]	uint8_t	Matriz que armazena que espaços de fichas estão usados ou não.

Todas as variáveis acima desempenham um papel essencial do projeto e se uma delas for removida, por afetar gravemente o resultado do jogo. Para o efeito, e a pedido no powerpoint de indicação do relatório, colocamos uma imagem das function calls na página seguinte. Devido à sua grande dimensão é possível analizar a imagem com maior facilidade com um zoom de 150% na página.

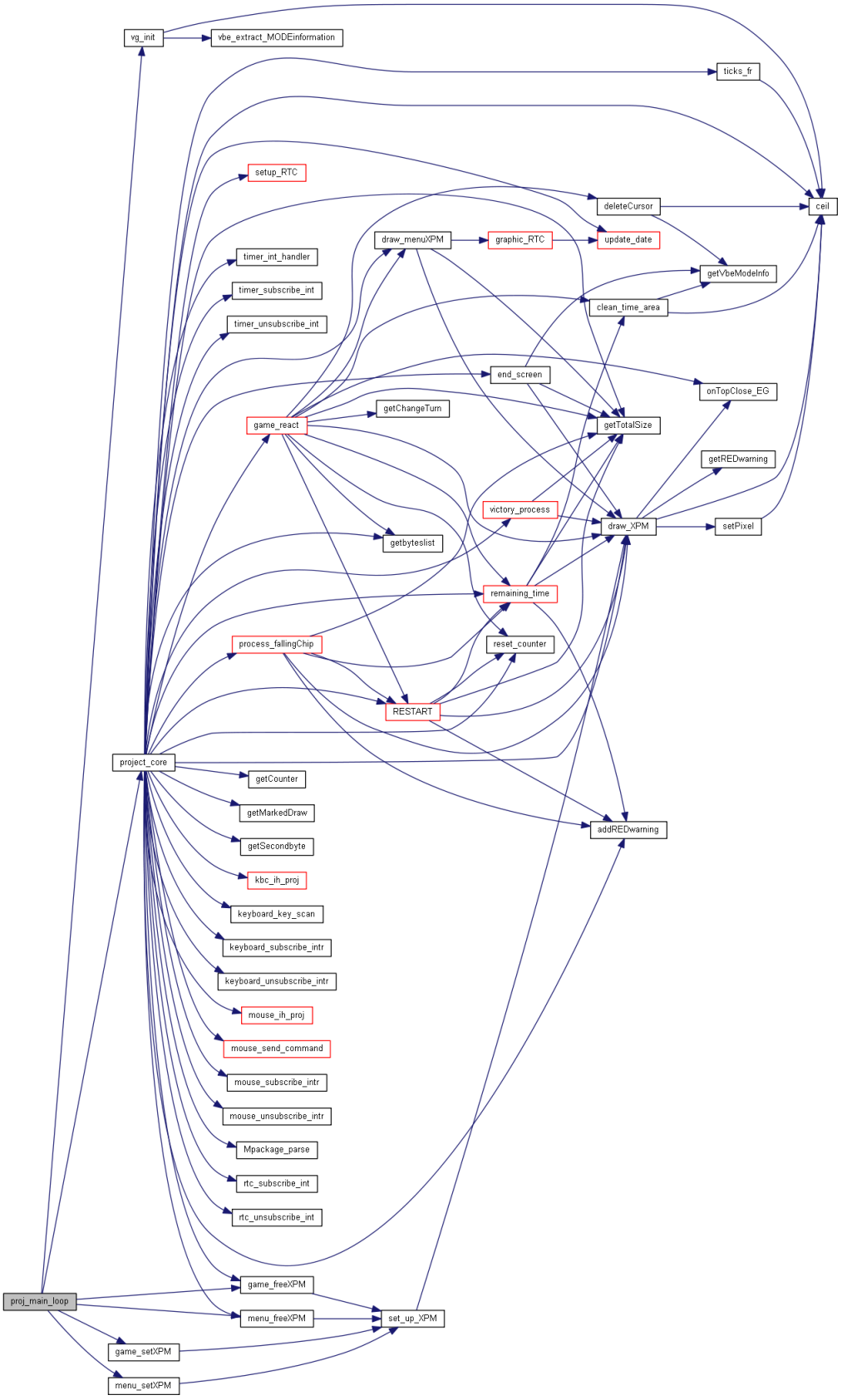


Figura 5: Mapeamento das Funções

7 Conclusões

Tendo terminado o nosso projeto, reconhecemos que as aulás práticas nos serviram de ajuda para o desenvolvimento do projeto. Todos os trabalhos laboratoriais realizados desempenharam um papel fundamental em entender os I/O devices de que fizemos uso e que puderam hoje ser compilados neste jogo. É essencial sublinhar que todos os trabalhos que fizemos no decorrer do semestre puderam ser reutilizados sem grandes modificações, o que acelarou bastante o desenvolvimento do jogo.

A aprendizagem de novos cuidados a ter na construção de qualquer código foi também um ponto importante. Tentativas de optimização e organização aperfeiçoaram a fluência do desenrolar do jogo e também nos fizeram entender o papel essencial da eficiência e redução de complexidade nos algoritmos e o quanto impacto isso pode ter em qualquer projeto.

Terminamos, assim, acreditando e defendendo que este projeto nos ajudará no futuro; seja no mercado de trabalho ou para cadeiras futuras. Finalmente, foi um prazer poder aprender mais e reconhecer a complexidade e composição dos devices que são usados com enorme regularidade nos nosso computadores.

Daniel Gonçalves
Guilherme Calassi