FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Relatório de Laboratório de Computadores - Projeto

2º Semestre

Ano letivo
2023/2024

Turma 10 - Grupo 4
Lucas Faria - up202207540
António Santos - up201907156
Rafael Campeão - up202207553
Alexandre Lopes - up202207015

Índice

Introdução

No âmbito do nosso projeto, optamos por desenvolver um jogo intitulado “Tennis for Two”. O propósito do jogo é vencer uma partida de ténis, sendo o primeiro jogador a marcar 10 pontos. A dinâmica do jogo envolve uma bola que deve ser rebatida com uma raquete, com dois jogadores posicionados em cada lado da quadra: um controlado pelo utilizador e o outro controlado pelo computador.

Durante o desenvolvimento do projeto, aplicamos várias das técnicas e dispositivos apresentados nas aulas de forma a criar a melhor experiência de jogo possível.

1.Instruções de utilização

1.1 Menu Inicial

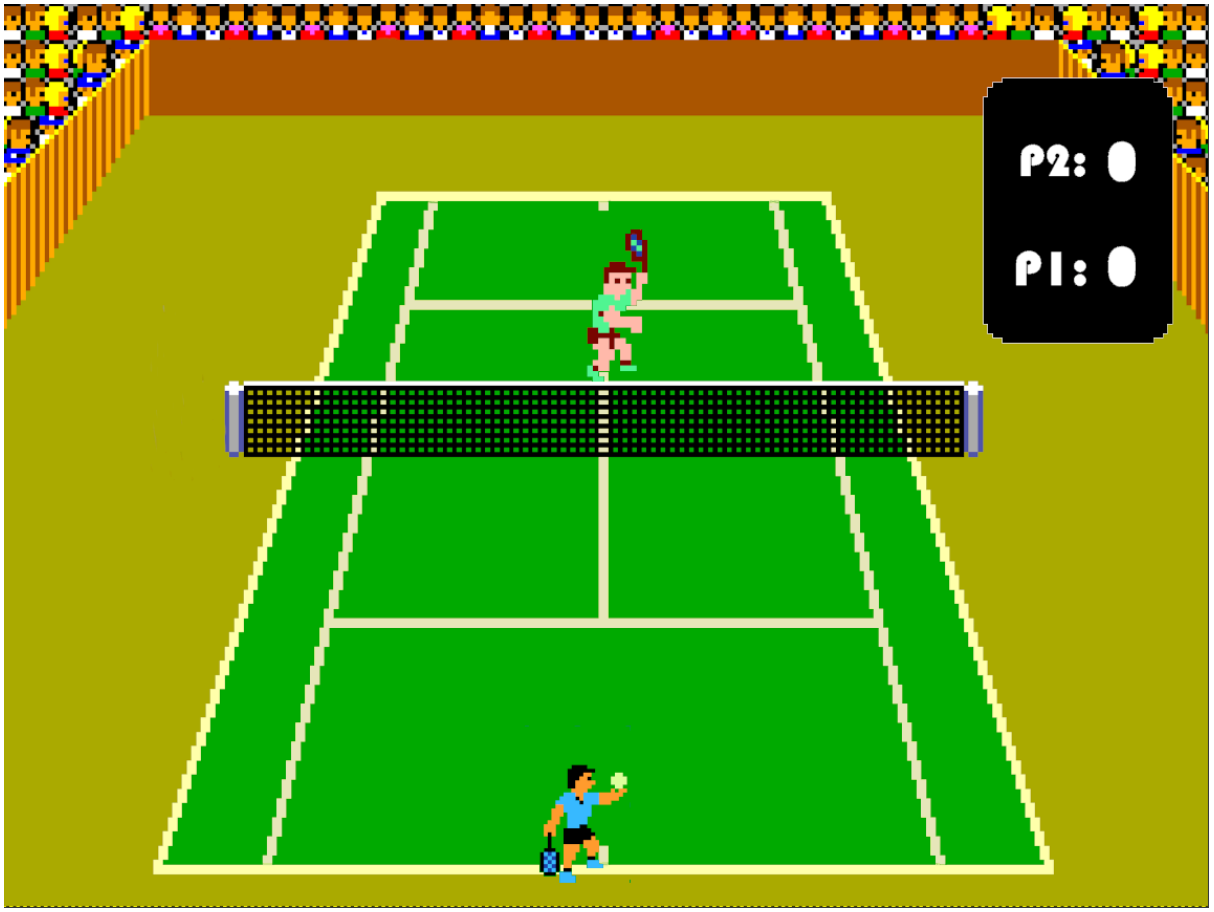
O menu inicial é o ponto de partida do jogo, onde o utilizador pode selecionar a opção de jogar ou sair do jogo.

Para navegar no menu, pode-se utilizar as teclas das setas ou o movimento do mouse. Para selecionar uma opção pode ser usada a tecla Enter ou o botão esquerdo do mouse.

Na parte inferior do menu, é possível observar a data atual.

1.2 Jogo

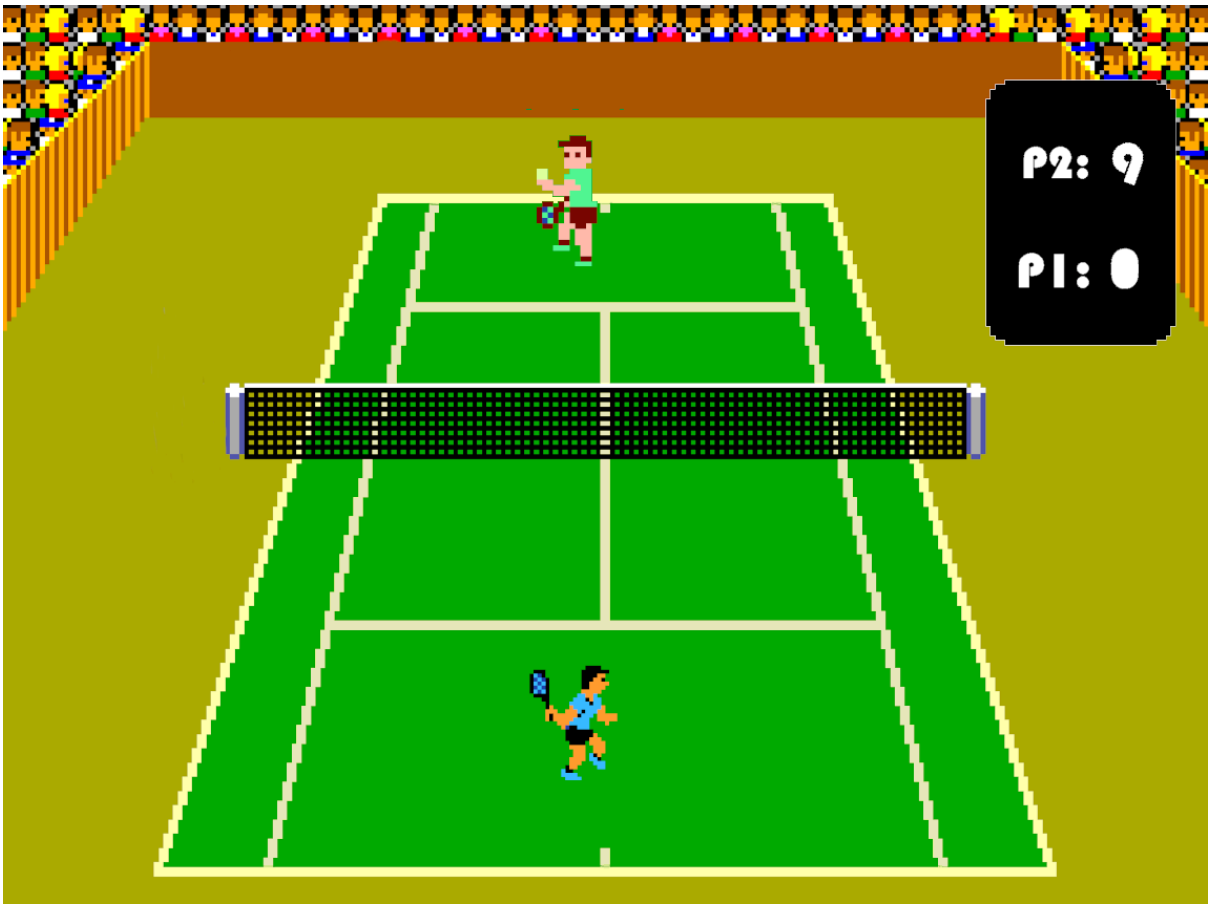
1.2.1 Serviço Jogador 1



O jogo começa com o utilizador a controlar a posição onde quer que o seu jogador faça o serviço para iniciar o jogo. Este mesmo procedimento ocorre quando o jogador marca algum ponto.

Para mover o jogador, podem ser usadas as teclas das setas esquerda e direita ou as teclas AD. Para realizar o serviço, o utilizador deve clicar no botão esquerdo do rato.

1.2.2 Serviço Jogador 2



Quando o jogador 2 marcar algum ponto, tem o direito a servir. Neste caso, o utilizador pode mover o seu jogador para onde quiser (mas não pode invadir o campo adversário).

Para mover o jogador, o utilizador deve usar as teclas das setas ou as teclas AWSD.

1.2.3 Bater na Bola e Movimento

Quando a bola está em jogo, o utilizador deve tentar marcar pontos contra o adversário. Para isso, deve mover-se para perto da bola e tentar acertar nela. É considerado ponto se a bola sair pelas linhas laterais do campo adversário. A pontuação atual é apresentada durante o jogo no lado direito do ecrã.

Para se movimentar, o utilizador deve usar as teclas das setas ou as teclas AWSD. Para acertar na bola deve premir o botão esquerdo do rato.

1.3 Menu de Pausa

Durante o jogo, se o utilizador premir a tecla ESC do teclado, o menu de pausa irá aparecer. Este menu tem três opções: retomar o jogo, reiniciar o jogo e sair. A primeira leva o utilizador de novo para o jogo que este pausou. A opção de reiniciar inicia um novo jogo. A opção de sair termina a execução do programa.

Para navegar no menu, o utilizador pode usar as teclas das setas ou o movimento do mouse. Para selecionar uma opção deve pressionar a tecla Enter do teclado ou o botão esquerdo do mouse.

2.Estado do Projeto

Dispositivo	Onde é usado	Interrupções
Timer	Controla os frames gerados, as animações e tempo para um certo evento acontecer	Sim
Keyboard	Movimento do jogador e controlo dos menus	Sim
Mouse	Controlo dos menus e usado para iniciar o movimento da raquete	Sim
Placa Gráfica	Gráficos do jogo e interface dos menus. Usada para verificar a colisão da bola com as linhas laterais	Não
RTC	Apresenta o dia no menu principal	Não
Serial Port	Não foi implementado (seria usado no multiplayer)	Não

2.1 Timer

O timer é usado para controlar a quantidade de frames gerados por segundo, atualizando o ecrã 60 vezes por segundo. Além disso, é usado para controlar a velocidade das animações e o momento em que certas ações devem acontecer.

A função ***timerHandler*** do módulo **game.c**, é executada a cada interrupção do timer, ou seja, a cada frame, quando o jogo está no estado ***GAME***.

Esta função é responsável por diversas tarefas essenciais:

- Desenhar o novo frame do jogo, incluindo o background, os jogadores e a bola
- Verificar e tratar colisões entre os elementos do jogo
- Controlar o movimento dos jogadores e da bola de acordo com as regras do jogo

- Atualizar a pontuação dos jogadores e exibi-la na interface do utilizador

Caso o estado atual seja ***START_MENU*** ou ***PAUSE_MENU***, a cada interrupção do timer o background será refrescado e a função ***timer_handler_menu*** do módulo **menu.c** será executada. Esta função é responsável por desenhar a interface do menu correspondente ao estado atual, incluindo a representação visual do rato.

Existe um counter que é incrementado pelo timer a cada interrupção e é usado por várias funções que controlam as animações e eventos do jogo. No módulo **player.c** o counter é usado na função ***updatePlayerMovementsTimer***, que atualiza as animações e a posição do jogador. Já no módulo **player2.c**, o counter é usado na função ***updatePlayer2AI***, que também atualiza as animações e movimentos, mas também para atrasar o jogador 2 (controlado pelo computador) quando este tem que fazer um serviço.

As funções responsáveis por controlar e programar o timer encontram-se no módulo **timer.c**.

2.2 Keyboard

O keyboard é usado para movimentar o jogador durante o jogo (teclas AWSD ou as setas), aceder ao menu de pausa (tecla ESC) e navegar nos menus (setas e Enter).

A cada interrupção do keyboard, uma função é executada de acordo com o estado atual do jogo:

Caso o estado atual seja ***GAME***, a função executada será a ***keyboardHandler*** do módulo **game.c.** Esta função chama a função ***changePlayerMovementKBD***, presente no módulo **player.c**, para atualizar o estado do movimento do jogador (se está parado ou em movimento, avaliando os makecodes e breakcodes recebidos) e a sua direção (dependendo do makecode recebido).

Caso o estado atual seja ***START_MENU*** ou ***PAUSE_MENU***, a função executada será a ***kbd_handler_menu*** do módulo **menu.c** para atualizar os botões selecionados nos menus e acioná-los (quando a tecla Enter é pressionada).

As funções que controlam e programam o keyboard encontram-se no módulo **kbd.c**.

2.3 Mouse

O mouse é usado para realizar a ação de movimento da raquete (para bater na bola usando o botão esquerdo) e para navegar nos menus. São usados os botões nos menus e no jogo, e os movimentos só nos menus.

Sempre que um novo pacote do rato é gerado, isto é, quando os 3 bytes que compõem um pacote do mouse são recebidos, caso o estado atual do jogo seja ***START_MENU*** ou ***PAUSE_MENU***, a função ***menuMouseHandler*** é executada. Esta função chama a função ***update_selected_mouse*** do módulo **menu.c** que lida com as interações entre o mouse e o menu, como selecionar os botões que estão na posição do mouse e acioná-los se o botão esquerdo for premido.

Independentemente do estado, a seguir, a função ***mouseHandler*** do módulo **game.c** é executada. Esta função chama outras funções que atualizam a posição do rato e verificam se o botão esquerdo foi clicado. A função que atualiza a posição é a **updateMousePosition**, definida no módulo **menu.c**, e a função que verifica se o botão esquerdo foi clicado é a ***updatePlayerMovementMouse***, definida no módulo **player.c**

As funções que controlam e programam o mouse encontram-se no módulo **mouse.c**.

2.4 Placa Gráfica

A placa gráfica é usada para mostrar os gráficos do jogo e a interface de utilizador dos menus. Foi usada a resolução 1152x864 e modo de cor direto. Além

disso, é usado double buffering com cópia para melhorar a experiência visual, evitando artefactos na imagem.

A placa gráfica é usada em várias funções de draw no programa.

A função ***drawBackground*** desenha a quadra do jogo e a função ***refreshBackground*** atualiza o frame do background, eliminando o frame anterior. Para além destas funções também existem a função ***drawDateBackground*** que desenha a data atual e a função ***drawElementsInMenuBackground***, usada para desenhar elementos específicos no fundo do menu. Estas funções encontram-se no módulo **background.c**.

A função ***drawPlayer*** desenha os jogadores (jogador 1 e 2) no ecrã. Esta função está definida no módulo **player.c**.

A função ***drawBall*** desenha a bola durante o jogo. Esta função está definida no módulo **ball.c**. Neste módulo também está definida a função ***checkCollisionLine*** que verifica se a bola ultrapassou alguma linha de jogo ao comparar as cores do local onde a bola será desenhada com as da linha lateral.

A função ***drawMouse*** desenha o ponteiro do mouse nos menus, enquanto as funções ***drawScore*** e ***drawScoreText*** desenham a informação que se encontra no placar de pontuações durante do jogo. Estas funções estão definidas no módulo **game_score.c.**

A função ***drawMouse*** desenha o ponteiro do mouse nos menus, enquanto as funções ***drawMenu*** e ***drawPause*** desenham toda a interface dos menus inicial e de pausa, respetivamente. Para desenhar os constituintes das interfaces, como os botões, estas duas funções utilizam a função ***draw_field*** que desenha um sprite na posição especificada. Estas funções estão definidas no módulo **menu.c**.

As funções que controlam e programam a placa gráfica encontram-se no módulo **video.c**.

2.5 RTC

O RTC é usado para apresentar a data atual no menu principal. A função ***get_date*** é usada para receber a data do rtc, e é invocada logo no início da função ***gameloop*** no módulo ***game.c*** e sempre que o jogo retorna ao menu principal.

As funções que controlam e programam o rtc encontram-se no módulo **rtc.c**.

2.6 Serial Port

Infelizmente, este dispositivo não foi implementado mas caso fosse, seria usado para o modo multiplayer do jogo permitindo que dois jogadores em computadores diferentes pudessem jogar ao mesmo tempo.

3.Estrutura do código

Módulo Timer: 7%
Módulo Keyboard: 7%
Módulo Mouse: 7%
Módulo Vídeo: 13%
Módulo RTC: 2%
Módulo Background: 5%
Módulo Ball: 10%
Módulo Game Score: 5%
Módulo Player: 10%
Módulo Player2: 10%
Módulo Menu: 10%
Módulo Game: 14%

3.1 Módulo Timer

Neste módulo estão implementadas funções que foram desenvolvidas no lab2 e que permitem programar o timer. As funções que eram desnecessárias para o projeto foram eliminadas.

3.2 Módulo Keyboard

Neste módulo estão implementadas funções que foram desenvolvidas durante o lab3 e que permitem programar o keyboard. As funções que eram desnecessárias para o projeto foram eliminadas.

3.3 Módulo Mouse

Neste módulo estão implementadas funções que foram desenvolvidas durante o lab4 e que permitem programar o mouse. As funções que eram desnecessárias para o projeto foram eliminadas.

3.4 Módulo Vídeo

Neste módulo estão implementadas funções que foram desenvolvidas durante o lab5 e que permitem programar a placa gráfica. Para além disso, também estão presentes neste módulo funções que permitem o funcionamento da técnica de double-buffering.

3.5 Módulo RTC

Neste módulo estão as funções que permitem programar o RTC. No caso do nosso programa, existe uma função que retorna o dia, mês e ano de forma direta, contudo é possível de forma indireta obter também os outros valores presentes no RTC (hora, minutos, …) através de uma função mais geral.

3.6 Módulo Background

Neste módulo estão as funções que permitem desenhar e “refrescar” o background do jogo (campo de jogo ou títulos e fundo preto nos menus).

3.7 Módulo Ball

Neste módulo encontra-se toda a lógica relacionada com a bola presente no jogo. O módulo contém a informação essencial da bola como a sua posição, declive/inclinação da sua trajetória, direção do movimento e um sprite com o

pixmap. Além disso, este módulo verifica as colisões da bola com a raquete do jogador e ajusta os valores da direção e de declive de acordo com a zona do impacto, verifica a colisão com as linhas laterais verificando se as cores da zona onde o novo frame da bola vai ser gerado tem as cores das linhas laterais, e ajusta a posição da bola tendo em conta a sua direção de movimento.

3.8 Módulo Game Score

Neste módulo encontra-se a lógica do placar de pontuações do jogo. O módulo escolhe os pixmaps adequados, tendo em conta o resultado do jogo. Os pixmaps dos resultados são atualizados neste módulo sempre que um jogador marcar algum ponto. Além disso, o módulo trata do desenho do placar de pontuações.

3.9 Módulo Player

Neste módulo encontra-se toda a lógica relacionada aos jogadores presentes no jogo. O módulo contém as informações essenciais de um jogador como a sua posição, direção de movimento, lado para onde está virado, estado (a mover, parado, a servir, a bater na bola, ..), o sprite atual em utilização e os conjuntos de sprites usados para as animações. Para além disso, o módulo contém funções para o desenho do jogador, controlo das ações e movimentos tendo em conta os inputs do utilizador (do keyboard e mouse) e controlo das animações.

3.10 Módulo Player2

Neste módulo encontra-se a lógica específica do jogador 2. O módulo controla as ações e movimentos do jogador 2. Este recebe as informações atuais do jogador 2 e da bola de forma a tomar as melhores decisões quando o jogador 2 é controlado pelo computador.

3.11 Módulo Menu

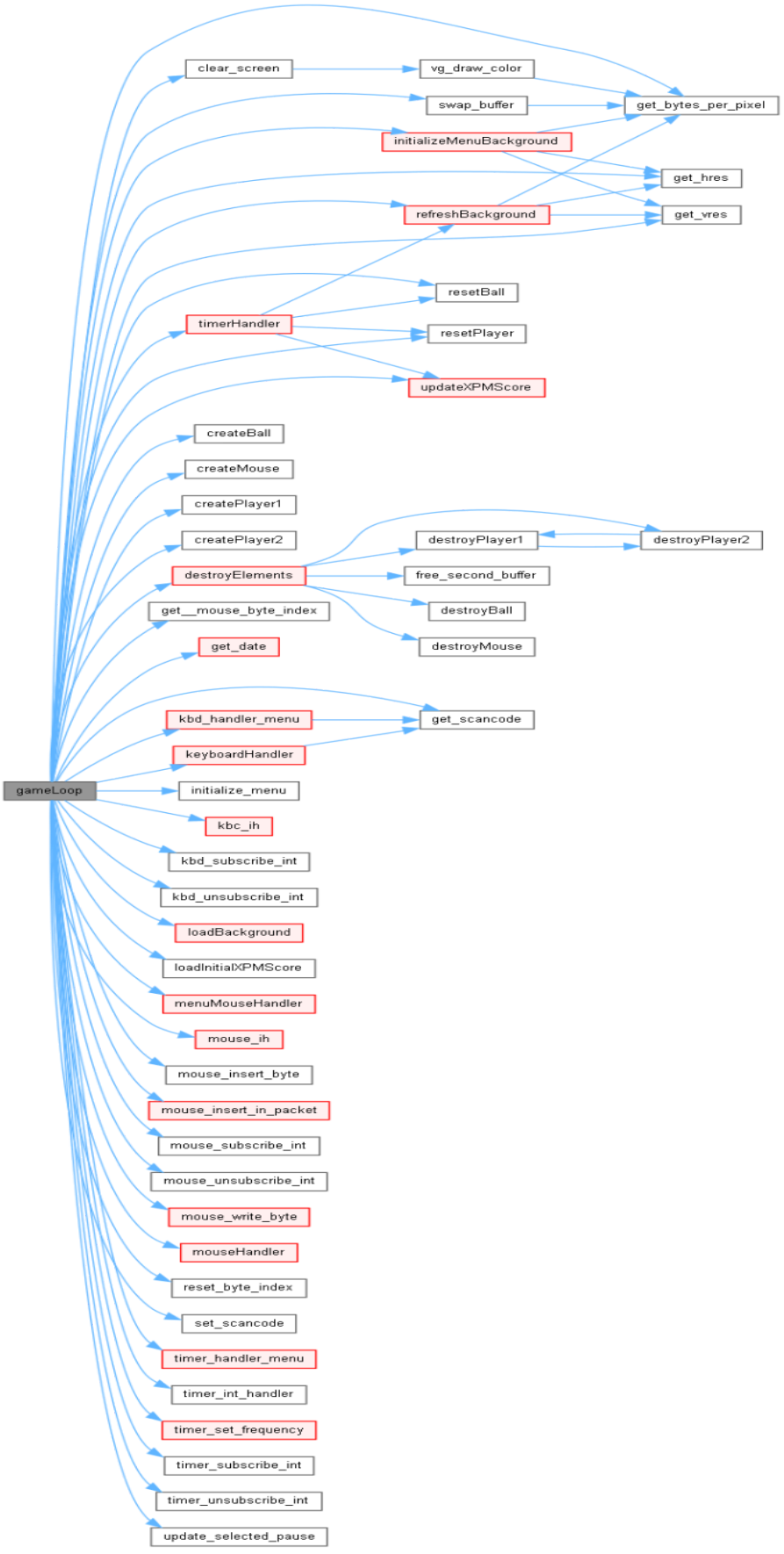
Neste módulo encontra-se a lógica para os menus. Este módulo contém todos os sprites necessários para cada menu e inicializa-os conforme necessário.

Também possui funções para desenhar estes menus e o rato e funções para controlar a sua navegação (sendo feita pelo rato ou pelo teclado).

3.12 Módulo Game

Este módulo é responsável por controlar a execução geral do programa. Ele inclui o ciclo com as interrupções (driver_receive) e trata das interrupções executando handlers correspondentes ao dispositivo que gerou a interrupção.

Function Call Graph



4.Detalhes da implementação

De forma a controlar o fluxo do jogo, utilizamos diversas **state machines** para nos ajudarem nessa tarefa. A principal é a **Game_state** que indica o estado do jogo, isto é, se estamos no menu inicial, a jogar, no menu de pausa, se vamos reiniciar o jogo ou sair deste. Além disso, os jogadores também têm state machines que indicam qual ação estão a executar (**Player_state** onde as ações são bater na bola, mover, …**),** qual a direção em que o jogador se está a mover (**Player_movement** onde as direções são cima, baixo, esquerda e direita) e em que direção o jogador está virado (**Player_direction** que pode ser direita ou esquerda). A bola também tem uma state machine que indica a sua direção de movimento (**Ball_direction** em que as direções são cima ou baixo).

Para dividirmos o código da melhor forma, usamos alguns conceitos de **programação orientada a objetos** como, por exemplo, usar variáveis estáticas que apenas podem ser acedidas através de getters e usar structs de forma a replicar um comportamento similar ao uso de classes em linguagens orientadas a objetos.

De forma a melhorarmos a performance, guardamos o background num buffer e quando precisamos gerar um novo frame, usamos a função **memcpy** para copiar o conteúdo do buffer da background para o buffer da placa gráfica. Desta forma, apagamos o frame anterior e o seu conteúdo. Depois deste passo, desenhamos os outros objetos “por cima”, isto é, os jogadores, a bola e o placar de pontuações. Esta estratégia evita que em cada frame tenhamos que desenhar todos os pixeís do zero, o que levaria a uma performance muito fraca visto que usamos um modo gráfico que usa muitos bytes (1152x864).

Além disso, para assegurar uma experiência visual mais fluida, usamos um **buffer secundário** que é preenchido antes do buffer principal. Isto assegura que não alteramos um frame enquanto este está a ser exibido, evitando assim algumas imperfeições visuais.

As **colisões** no jogo são detectadas com o uso das coordenadas dos objetos. Contudo, no caso da bola, também é usado o conteúdo do buffer da placa gráfica para verificar se na nova posição onde a bola será desenhada, a cor de fundo é

igual à das linhas laterais. Dessa forma podemos de uma forma eficiente determinar se a bola saiu fora do campo.

Para melhorar a resposta do mouse e diminuir o tempo de resposta, foi necessário diminuir o **sample rate** do valor padrão (200) para um valor inferior (40). Desta forma, o programa consegue processar de forma mais eficaz os pacotes que recebe do rato sem qualquer atraso nos comandos. Este comando do mouse não foi discutido nas teóricas nem nas práticas. Seria bastante interessante que este tópico pudesse ser discutido no futuro visto que pode ser útil para projetos que dependam dos movimentos do mouse em "tempo real" com baixo atraso nos comandos.

O **RTC** foi implementado de forma a obter os dados do dia, mês e ano atuais. Estes dados são usados no menu principal para apresentá-los ao utilizador. É de notar que usamos a **Update in progress flag (UIP)** presente no registo A do RTC para verificar se o RTC está a atualizar. Caso este esteja em atualização, não lemos o valor para evitar apresentar dados errados ao utilizador.

Apesar de no **Project Specification** termos planeado usar o **RTC** para "missões diárias" que iriam recompensar o utilizador por jogar diariamente, decidimos antes apresentar a informação de data no menu pois chegamos à conclusão que a proposta inicial não faria muito sentido para o tipo de jogo que estamos a desenvolver. Além disso, não nos foi possível implementar a funcionalidade de Multiplayer que usaria a **serial port.**

5.Conclusão

Este projeto foi muito enriquecedor para a nossa aprendizagem nesta unidade curricular. Para além de nos ter ajudado a melhorar as nossas habilidades de programação em C, também nos ajudou a entender melhor como é que os periféricos funcionam.

Apesar de termos tido alguns problemas como lentidão no programa, devido ao peso das funções de desenho dos pixmaps, e lentidão na resposta aos

movimentos do rato, podemos considerar que no geral o projeto correu bem e permitiu-nos aplicar os conhecimentos aprendidos ao longo do semestre.

Infelizmente, não conseguimos implementar a porta de série devido a constrangimentos de tempo e trabalho para outras unidades curriculares mas é uma funcionalidade que no futuro gostaríamos de incluir no nosso programa.

Notas

Link para o vídeo de demonstração: https://www.youtube.com/watch?v=EUo9TWKAHkw