Ultimate Open 2020

Relatório final do projeto de LCOM

Turma 3, Grupo 4
Maria José Valente da Silva Carneiro (up201907726)
Rodrigo Tuna de Andrade (up201904967)

Janeiro 2021

Conteúdo

1 Introdução

Este projeto, com o título de "Ultimate Open 2020"foi realizado como projeto final da unidade curricular Laboratório de Computadores com objetivo de aplicar de forma mais elaborada e complexa os diferentes dispositivos lecionados ao longo do semestre.
O projeto trata-se de um video-jogo de ténis com diferentes modos de jogo e foi realizado tendo em conta os objetivos da unidade curricular:

- Utilizar interfaces de hardware mais comum em periféricos de computador.
- Desenvolver software de baixo nível.
- Programar na linguagem C (de forma estruturada).
- Utilizar diferentes ferramentas de desenvolvimento de software.

2 Instruções de Utilização

2.1 Menu Inicial

Ao iniciar o programa, é apresentado o menu inicial que permite ao utilizador escolher entre 4 opções:

- **Singleplayer:** Inicia o jogo no modo singleplayer.
- **Multiplayer:** Redireciona a um menu de escolha entre criar um jogo no modo multiplayer ou participar num criado previamente, iniciando o jogo a partir desse momento.
- **Scores:** Apresenta os 4 melhores resultados obtidos pelos jogadores no modo singleplayer, bem como o seu nome, data e hora em que foram alcançados.
- **Quit:** Termina o programa, tal como pressionando a tecla **ESC**.

Cada uma das opções neste menu e nos menus subsequentes pode ser selecionada pressionando o botão esquerdo do rato sobre cada uma delas.

Figura 1: Menu Inicial

2.2 Modo Singleplayer

O objetivo do jogo no modo Singleplayer é obter o maior número de pontos possíveis ao atingir a bola. Esta é lançada de um ponto fixo (máquina) em diferentes direções, aumentando a sua velocidade ao longo do tempo.
O jogador consegue rematar a bola caso esta se encontre na proximidade da raquete e caso pressione o botão esquerdo do mouse. A bola é enviada para o

ponto onde se encontra a mira, que segue o deslocamento do rato. A raquete troca de lado (esquerda para a direita, ou vice-versa) consoante a posição relativa da bola ao jogador.
De modo a controlar o movimento do personagem, são usadas as teclas **W**, **A**, **S**, **D** do teclado, que o permitem mover-se nas 8 direções dos pontos cardeais e colaterais.
Ao longo do jogo, a pontuação é incrementada a cada remate bem sucedido no canto superior esquerdo do ecrã.
O jogo termina quando o utilizador não consegue atingir/rematar a bola, quando a manda para fora das linhas de campo ou para a sua própria área de jogo.

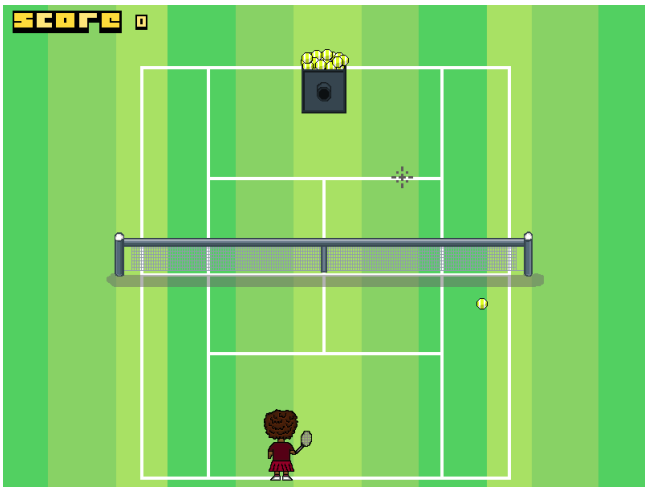


Figura 2: Modo Singleplayer

Após o jogo ter terminado ou no caso de desistência (pressionando a tecla **ESC** a qualquer ponto do jogo), o utilizador é redirecionado para um menu de gameover ou para um menu que lhe permite inserir o seu nome, se a pontuação obtida for um highscore.

2.3 Scoreboard

Neste menu, o utilizador consegue verificar as 4 pontuações mais altas obtidas no jogo no modo singleplayer. Cada entrada na tabela contém o nome escolhido pelo jogador após ter obtido um highscore, a hora e a data em que o obteve, e a pontuação que atingiu. Para regressar ao menu principal, é necessário pressionar o botão esquerdo do rato sobre o botão **HOME**.

Figura 3: Scoreboard

A informação da scoreboard é atualizada sempre que, no modo singleplayer, um jogador obtém uma pontuação superior às registadas anteriormente. Caso isso aconteça, após terminar o jogo e antes de surgir o menu de gameover, é pedido ao utilizador que preencha o seu nome de modo a ser identificado na scoreboard. Quando estiver satisfeito com o nome escolhido (que tem um máximo de 10 caracteres), é necessário pressionar o botão esquerdo do rato sobre o botão **DONE** para que o nome seja atribuído à pontuação que obteve.

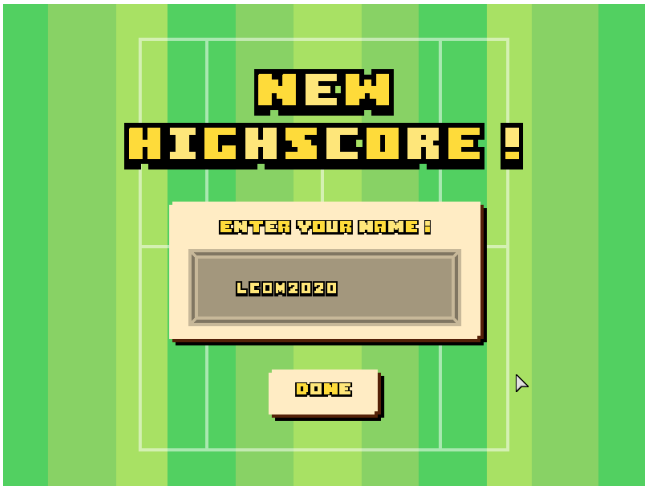


Figura 4: Preenchimento do nome do jogador num novo highscore

2.4 Modo Multiplayer

Antes do jogo ser iniciado, o utilizador é direcionado a um menu que lhe permite escolher entre criar um jogo neste modo ou de se juntar a um começado previamente. O utilizador que seleciona o botão **CREATE** joga no campo inferior e o que escolhe a opção **JOIN** no campo superior. Para retornar ao menu principal a partir deste menu, basta pressionar a tecla **ESC**.

Figura 5: Seleção entre criar um jogo ou juntar-se a um já criado

Enquanto os jogadores aguardam para que o jogo se inicie e, em particular, o jogador que o cria pelo seu adversário, é mostrada uma mensagem de espera. Caso nenhum jogador se junte à partida em pelo menos 30 segundos, o utilizador é redirecionado para o menu de escolha. Também pode voltar ao menu anterior voluntariamente pressionando a tecla **ESC**.

(a) Mensagem do jogador que cria o jogo

(b) Mensagem do jogador que se junta ao jogo

Figura 6: Mensagens de espera

Após ambos os jogadores estarem conectados, o jogo é iniciado. O objetivo do jogo no modo multiplayer é conseguir vencer uma partida disputada à melhor de 4 pontos.
A partida começa com o serviço feito pelo jogador que cria o jogo e que se movimenta pelo campo inferior. No momento do serviço, o jogador que o efetua não se consegue mover livremente pelo seu campo. Caso algum dos jogadores não consiga atingir/rematar a bola, a mande para fora das linhas de campo ou para a sua própria área de jogo, o adversário ganha um ponto e efetua o próximo serviço.
Tal como no modo singleplayer, o jogador consegue rematar a bola caso esta se encontre na proximidade da raquete e caso pressione o botão esquerdo do mouse. A bola é enviada para o ponto onde se encontra a mira, que segue o deslocamento do rato. A raquete troca de lado (esquerda para a direita, ou vice-versa) consoante a posição relativa da bola ao jogador.
Os pontos de cada jogador são modificados ao longo do jogo, no canto superior esquerdo e são conhecidos como 15 (1 ponto), 30 (2 pontos), 40 (3 pontos) e o quarto ponto resulta no ponto vencedor da partida, que a termina. Caso a pontuação esteja a 40-40, ganha o jogador que consiga obter 2 pontos primeiro.
Após o jogo ter terminado, cada utilizador é redirecionado para um menu de gameover.

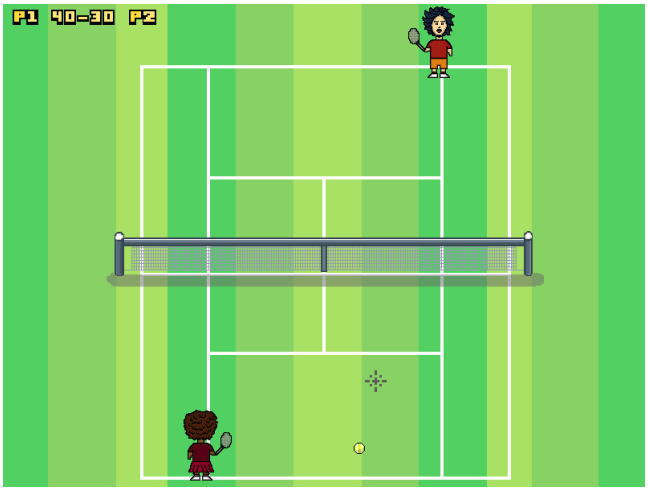


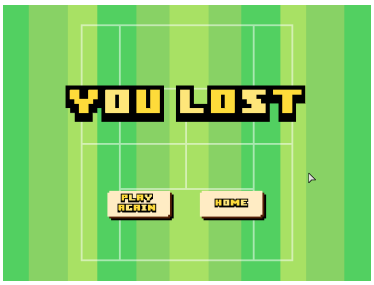
Figura 7: Modo Multiplayer

2.5 Gameover

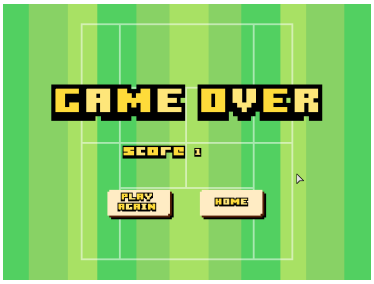
Tanto no caso do jogo ser no modo singleplayer ou no modo multiplayer, após cada partida terminar, o utilizador é redirecionado para o menu de gameover. A partir deste menu, em ambos os casos, tem a opção de jogar de novo, es-

colhendo o botão **PLAYAGAIN**, que o redireciona para o menu de escolha, no caso do modo ser multiplayer, ou para o próprio jogo no caso do modo ser singleplayer. Também é dada a opção de retornar ao menu inicial, escolhendo o botão **HOME**.
Se o jogo for no modo singleplayer, é mostrada uma mensagem de gameover e a pontuação obtida, e se for no modo multiplayer é mostrada uma mensagem que indica ao utilizador se este ganhou ou perdeu a partida.

(a) Mensagem de vitória no modo multiplayer



(b) Mensagem de derrota no modo multiplayer



(c) Mensagem no modo singleplayer

Figura 8: Diferentes mensagems de gameover

3 Periféricos Utilizados

Os periféricos utilizados no projeto estão descritos na seguinte tabela:

Periférico	Função	Interrupções
Timer	Controlo da cadência de frames e gerir eventos do jogo	sim
Keyboard	Controlo do movimento dos jogadores e preenchimento de campos	sim
Mouse	Controlo do remate da bola e navegação pelos menus	sim
Video Card	Apresentação do ecrã	não
RTC	Apresentação da data e da hora e alarmes	sim
UART	Comunicação entre computadores no modo multiplayer	sim(a receber)

3.1 Timer

O timer é utilizado para regular a frame rate do jogo. Utilizando a sua frequência pré - definida de 60 Hz, a cada interrupção são atualizadas as posições das diferentes entidades e a cada 2 interrupções é redesenhado o ecrã. Também a cada interrupção são verificadas condições que possam levar a alteração no estado do programa, como a de a bola estar fora dos limites do ecrã.
A implementação do timer foi realizada nos ficheiros **i8254.h** e **timer.c**.

3.2 Keyboard

O teclado é utilizado para controlar o movimento dos jogadores, alterando a sua velocidade. O jogador tem uma velocidade com duas componentes x e y, que são alteradas quando é recebido o primeiro makecode de uma tecla reservada ao movimento (W,A,S,D) ou um breakcode das mesmas teclas. Isto permite um movimento fluido e que o jogador se possa movimentar nas diagonais. O teclado é também utilizado no preenchimento de campos, como o nome do jogador quando realiza um highscore. São utilizados os makecodes das teclas que representam numeros e letras para escrever e também a tecla de apagar. Cada interrupção do teclado causa a leitura de um byte.
A implementação do keyboard foi realizada nos ficheiros **i8042.h**, **kbc.c**, **kbc.h**, **keyboard.h** e **keyboard.c**.

3.3 Mouse

O mouse é utilizado na navegação dos menus, controlando um cursor que se movimenta consoante o deslocamento do rato e, para selecionar uma opção, é necessário pressionar o seu botão esquerdo. Similarmente, o rato é utilizado na ação de rematar, controlando uma mira que indica onde a bola irá ser rematada, e, para efetuar o remate, é utilizado também o botão esquerdo do rato.
Utilizamos o modo Data Reporting do rato e a cada interrupção é lido um byte do buffer. Após serem lidos 3 bytes é formado um packet com a ação do rato.
A implementação do mouse foi realizada nos ficheiros **kbc.c**, **kbc.h**, **mouse.h** e **mouse.c** e **mouse_macros.h**.

3.4 Video Card

A video card é utilizada para apresentar a informação gráfica do jogo. É utilizado o modo gráfico 0x115 com uma resolução 800x600 e 24(8-8-8) bits por pixel com direct color mode. Para carregar ficheiros do tipo xpm foi utilizada a função fornecida pela lcf *xpm_load*, que depois são usados em sprites e na animações deles. Todos os ficheiros xpm utilizados são da nossa autoria. De forma a a garantir a suavidade da imagem foi implementada a função de double buffering com page flipping que troca de buffer com recurso a chamadas da função *07h-Set/Get Display Start* da VBE.
A implementação da video card foi feita nos ficheiros **video_macros.h**, **video.h** e **video.c**.

3.5 RTC

O RTC (Real Time Clock) é utilizado para ler a data e a hora atual, quando o jogador apresenta um novo highscore, de modo a poder guardar essa informação na scoreboard. Usamos também as suas interrupções de alarme para controlar o tempo de espera do jogador que cria um jogo, no modo multiplayer: uma interrupção de alarme é gerada 30 segundos após criar o jogo e se nenhum outro jogador se juntar, regressa ao menu anterior. Os detalhes específicos de implementação do RTC são discutidos na secção *Detalhes de Implementação*.
A implementação do RTC foi feita nos ficheiros **rtc_macros.h**, **rtc.h** e **rtc.c**.

3.6 UART

A UART é utilizada na comunicação entre computadores distintos, necessária ao modo multiplayer e configurada com 8 bits por caracter, 2 stop-bits e erro de paridade par. Foram utilizadas as interrupções de receção e erro, bem como FIFO's com um aviso the trigger de 8 bytes. O envio de mensagens é feito através de poll do Transmiter Holding Register, colocando um caracter quando este se encontra vazio. A receção de mensagens utiliza interrupções, sendo que a cada interrupção são lidos todos os caracteres presentes na FIFO e colocados numa queue de receção para posteriormente serem tratados. Os detalhes da comunicação específicos ao projeto são explorados na secção de *Detalhes de Implementação*.
A implementação da UART foi feita nos ficheiros **uart_macros.h**, **uart.h** e **uart.c**.

4 Organização e Estrutura do código

4.1 Módulos de Periféricos

4.1.1 timer.c

Este módulo contém as funções necessárias à utilização do timer e foi desenvolvido durante o lab2.
Peso: 2%
Responsável: Maria Carneiro e Rodrigo Tuna (50/50)

4.1.2 kbc.c

Este módulo contém as funções necessárias à utilização do kbc e foi desenvolvido durante o lab3.
Peso: 1%
Responsável: Maria Carneiro e Rodrigo Tuna (50/50)

4.1.3 keyboard.c

Este módulo contém as funções necessárias à utilização do teclado e foi desenvolvido durante o lab3.
Peso: 2%
Responsável: Maria Carneiro e Rodrigo Tuna (50/50)

4.1.4 mouse.c

Este módulo contém as funções necessárias à utilização do rato e foi desenvolvido durante o lab4 e adicionada a máquina de estados para o botão esquerdo.
Peso: 5%
Responsável: Maria Carneiro e Rodrigo Tuna (50/50)

4.1.5 video.c

Este módulo contém as funções necessárias à utilização da placa gráfica e foi desenvolvido durante o lab5.
Peso: 5%
Responsável: Maria Carneiro e Rodrigo Tuna (50/50)

4.1.6 rtc.c

Este módulo contém as funções necessárias à utilização do rtc.
Peso: 7%
Responsável: Maria Carneiro

4.1.7 uart.c

Este módulo contém as funções necessárias à utilização da uart.
Peso: 8%
Responsável: Rodrigo Tuna

4.2 Módulos do jogo

4.2.1 communication.c

Este módulo contém as funções de envio e tratamento de caracteres recebidos pela uart.
Peso: 5%
Responsável: Rodrigo Tuna

4.2.2 drivers.c

Este módulo contém as funções de subscrição, configuração e des-subscrição de todos os periféricos do projeto, assim como a função que recebe interrupções.
Peso: 1%
Responsável: Rodrigo Tuna

4.2.3 entities.c

Este módulo contém as funções utilizadas no movimento das entidades presentes no jogo. Contém também as estruturas *ball* e *player*.
Peso: 5%
Responsável: Maria Carneiro e Rodrigo Tuna (40/60)

4.2.4 game.c

Este módulo contém as funções de implementação dos diferentes modos de jogo.
Peso: 18%
Responsável: Maria Carneiro e Rodrigo Tuna (10/90)

4.2.5 gameLogic.c

Este módulo contém as funções relacionadas com a lógica do jogo, como deteção de jogadas inválidas, contagem dos pontos ou a alteração do estado do jogo.
Peso: 5%
Responsável: Rodrigo Tuna

4.2.6 menu.c

Este módulo contém as funções de implementação dos diferentes menus do programa. Peso: 22%
Responsável: Maria Carneiro e Rodrigo Tuna (80/20)

4.2.7 sprite.c

Este módulo contém as implementação dos sprites e animação dos mesmos. Contém também as estruturas *sprite* e *animated_sprite*.
Peso: 8%
Responsável: Maria Carneiro

4.2.8 scoreboard.c

Este módulo contém as funções de implementação da apresentação dos highscores. Peso: 5%
Responsável: Maria Carneiro

4.2.9 utilities.c

Este módulo contém as funções de utilidade ao projeto assim como a implementação da estrutura *queue*.
Peso: 1%
Responsável: Rodrigo Tuna

4.3 Gráfico de chamada de funções

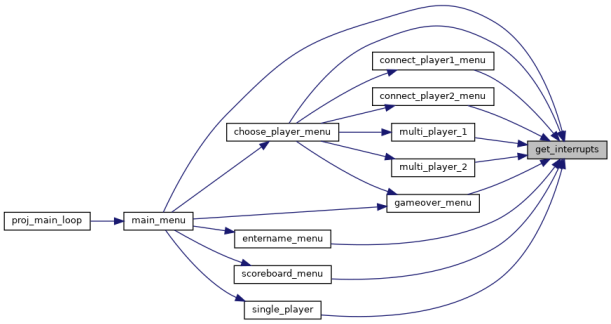


Figura 9: Gráfico de chamada de funções

Neste gráfico é possível observar as principais funções do programa que são os menus, os modos de jogo e a função *get_interrupts* que é a única função que chama a função *driver_receive*

5 Detalhes de implementação

5.1 Máquinas de Estado

Foram implementadas máquinas de estado para diferentes objetivos: controlo do movimento do jogador, deteção do click do botão esquerdo do rato e para os menus.
A máquina de estado para o controlo do jogador tem como função alterar a velocidade apenas quando há uma alteração no estado da tecla, seja primir ou levantá-la, visto que primir uma tecla pode gerar varios makecodes e nem todos esses códigos alteram a velocidade da personagem.
A máquina de estado para o rato guarda o estado do botão esquerdo sendo que apenas retorna o evento de left-click quando há uma variação no estado e este passa a ser de primido, e não sempre que este butão se encontra primido.
A máquina de estado dos menus, guarda o estado em que o menu se encontra, que representa a opção em que o utilizador colocou o rato sobre, este estado é alterado pela ação do movimento do rato.

5.2 Animação de Sprites

Ao longo do jogo, recorremos à animação de alguns sprites: os botões em todos os menus, "descem"quando o cursor é colocado sobre eles, de modo a simular o movimento de pressionar um botão real; os jogadores aparentam agachar e as suas raquetes trocam de lado consoante a posição relativa da bola ao jogador. Estes casos necessitavam de características específicas para serem animados, logo decidimos criar uma struct individual que representásse este tipo de objetos (**animated_sprite**). Esta struct engloba um **sprite**, variáveis intrínsecas ao processo de animação e um array de pixmaps que representam a sequência de frames a serem apresentados.
O processo de animação está em maior parte na função **update_sprite_animation**, que, quando o delay associado à animação é atingido, troca a imagem a ser mostrada dentro de um set selecionado.
Para o caso dos menus, como a animação apenas surge caso o cursor esteja em certas partes do ecrã, o delay é 0 dado que não é necessário uma sequência de imagens ser alternada ao longo do tempo e o número de sets depende apenas do número de botões utilizados.
Já no caso dos jogadores, o delay é 30 dado que existe uma sequência de imagens a ser alternada ao longo do tempo e o número de sets depende do número de posições diferentes dos personagens (se estão agachados ou não, com a raquete à esquerda ou à direita (**change_racket_sprite**)).
A atualização das animações é feita a cada interrupção recebida pelo timer.

5.3 RTC

De modo a aproveitar as funcionalidades do RTC, escolhemos não só usá-lo para ler a data e a hora atual, mas também para gerar interrupções de alarme.

Para isso efetuamos a subscrição das suas interrupções (**rtc_subscribe_int**) e a ativação de interrupções de alarme no início do programa (**init_all**), ativando o bit 5 do registo de controlo B (**rtc_enable_int**). No fim do programa (**reset_all**), desativamos as interrupções de alarme (**rtc_disable_int**) bem como desubscrevemos as interrupções do RTC (**rtc_unsubscribe_int**). Apesar de termos usado para o caso de interrupções de alarme, as funções de ativação/desativação de interrupções que críamos permitem também a ativação ou desativação de interrupções periódicas e de update.
No contexto do nosso projeto, as interrupções de alarme pareceram-nos as mais proveitosas para conseguirmos receber apenas uma interrupção quando a necessitamos, cujo é no momento de espera para conexão do outro jogador no modo multiplayer. Para isso, criamos uma função **rtc_set_alarm** que programa um alarme para um certo número de segundos após a hora atual. A função **rtc_ih** altera uma variável booleana global que indica se recebeu uma interrupção de alarme, após a leitura do registo de controlo C e a verificação se o bit 5 está ativo. Caso sejam emitidos outros tipos de interrupções do RTC, essas são ignoradas.
Sempre que lemos algum valor de qualquer registo de data ou hora, verificamos se o RTC tem alguma atualização em curso a partir da leitura do registo de controlo A e, caso o bit 7 esteja ativo, a leitura da data ou hora não é efetuada (**rtc_read_date**). Para agilizar o processo de leitura da data e hora do RTC, criamos a função (**rtc_get_date**) que lê a informação necessária dos respetivos registos e guarda-a num array.

5.4 Protocolo de comunicação

Antes que se possa começar o jogo multiplayer é necessário fazer a sincronização dos dois computadores. Essa sincronização é feita da seguinte forma: no menu do jogo multiplayer existem as opções **CREATE GAME** e **JOIN GAME**, ao ser selecionada a opção **CREATE GAME**(que deve ser selecionada primeiro) é atribuido o jogador 1 e o programa passa a uma posição passiva, procurando receber o caracter '2', contrariamente a opção **JOIN GAME** faz com que seja atribuido ao utilizador o jogador 2 e o programa começa por adotar uma posição ativa enviando o caracter '2'. Após esta primeira fase são trocados os processos, passando o jogador 1 para ativo e enviando ao caracter '1' e o jogador para passivo procurando receber o caracter '1'. Após a sincronização o jogo é iniciado.
Durante o jogo existem dois tipos de mensagens possíveis de ser enviados, mensagens referentes ao jogador ou mensagens referentes à bola.
As mensagens referentes ao jogador têm 2 bytes sendo o primeiro o caracter 'P' e o segundo o make ou break code da tecla primida pelo utilizador. Estas mensagens são enviadas sempre que há uma interrupção do teclado num computador e é utilizada replicação de máquina de estados para garantir a igual posição dos jogadores nos dois computadores.
As mensagens referentes à bola têm 3 bytes em que o primeiro é a concatenação dos bytes mais significativos das posições x e y para onde a bola será enviada, como o valor mais alto para a posição da bola é 800 então o byte mais signifi-

cativo da posição terá no máximo 3 bits pelo que é possível fazer esta junção utilizando operações bit a bit. Os seguintes bytes são os bytes menos significativos das posições x e y. Este tipo de mensagem é enviado sempre que um jogador é capaz de acertar na bola, fazendo com que esta altere a sua trajetória sendo por isso necessário que a trajetória da bola seja alterada nos dois computadores.

5.5 Deteção de Colisões

Foi utilizada a deteção de colisão para a ação de rematar, sendo que a trajetória da bola só é alterada se no momento em que é primido o botão esquerdo do rato, a bola se encontar dentro de uma hit-box localizada na zona da raquete do jogador. Foi também utilizada a deteção de colisão entre os sprites e os limites do ecrã ou os limites definidos para a sua movimentação, não permitindo ao sprite sair dessa zona.

6 Conclusão

Com a realização do projeto sentimos que foram atingidos os objetivos da unidade curricular. O trabalho permitiu-nos ganhar um conhecimento mais aprofundado acerca dos periféricos do programa devido a todos os problemas que encontramos durante a sua realização e que não encontramos durante os labs, assim como a implementação de novos periféricos que não tínhamos testado antes.
A unidade curricular foi bastante exigente, especialmente no início devido à matéria ser de difícil compreensão, sendo que os primeiros labs foram realizados com bastante ajuda por parte do professor das aulas práticas e do monitor. Mas à medida que o semestre foi passando, os labs foram feitos com maior facilidade, sendo que a implementação do RTC foi realizada sem grandes dificuldades, apesar de não se poder dizer o mesmo da UART.
Como aspetos positivos gostaríamos de destacar todo o acompanhamento que tivemos por parte dos professores, tanto nas aulas práticas como via email ou teams, e do monitor. Achamos que o método de trabalho a pares é uma mais-valia para a compreensão da matéria pois permite que os colegas de trabalho se ajudem mutuamente.
Como aspetos negativos salientamos as duas provas de programação, visto que em ambas o enunciado continha erros e também a demora em entregar as notas das mesmas provas. Concluindo, a apreciação geral da unidade curricular é positiva contudo a avaliação é um aspeto a melhorar.