MASTERMINIX

Turma 6 - Grupo 2

Realizado por:

- Afonso Osório (up202108700)
- António Silva (up202108760)
- Daniel Carneiro (up202108832)
- Gonçalo Costa (up202108814)

Índice

Introdução

Masterminix, inspirado no Mastermind, é um jogo enigmático de tabuleiro disputado entre dois jogadores.
O primeiro jogador ficará encarregue de adivinhar o código colorido introduzido pelo outro jogador, tendo 9 tentativas para o fazer.
Já o segundo jogador introduziu o código vencedor, bem como dará pistas ao primeiro, caso ele tenha adivinhado uma das cores da sequência, bem como a sua posição.
O jogo acabará no momento em que o primeiro jogador introduzir a sequência vencedora.

Instruções

Menu Inicial

Iniciando o Masterminix, os jogadores deparam-se com um menu principal com dois botões, “START” para iniciar o jogo e “END”.

Fig.1 - Menu inicial

Tabuleiros de jogo

Após clicar no botão “START” poderão aparecer dois tabuleiros de jogo diferentes.
No caso do jogador 1 que terá de adivinhar a solução, o seu tabuleiro terá 9 filas onde poderá introduzir a possível solução por turnos e 8 cores a introduzir no tabuleiro. As bolas serão colocadas com o botão esquerdo do rato e retiradas com o botão direito do rato.

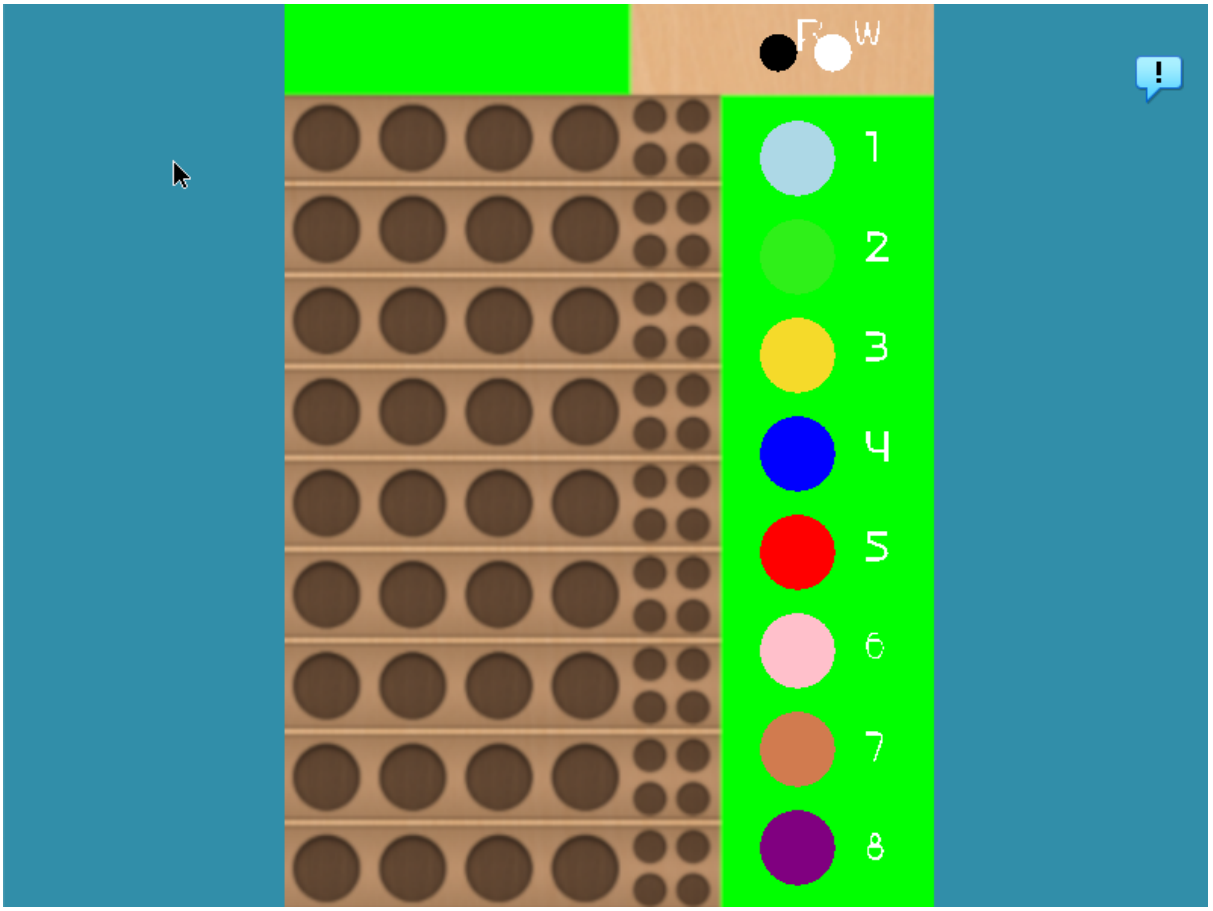


Fig.2 - Tabuleiro do jogador 1, após clicar no “START”

Tabuleiros de Jogo

No caso do jogador 2, podemos encontrar algumas diferenças em relação ao tabuleiro anterior.
Encontramos mais uma fila de jogo, onde o utilizador introduzirá a chave de jogo e vai poder usar as bolas pretas, caso o jogador 1 tenha introduzido a bola de uma cor no sítio certo da solução e brancas, caso o jogador 1 tenha introduzido uma bola de uma das cores da solução mas no sítio errado de forma a confirmar as jogadas. Se a bola e a cor não corresponderem ao local onde a bola foi colocada, o jogador 2 não colocará nenhuma bola.

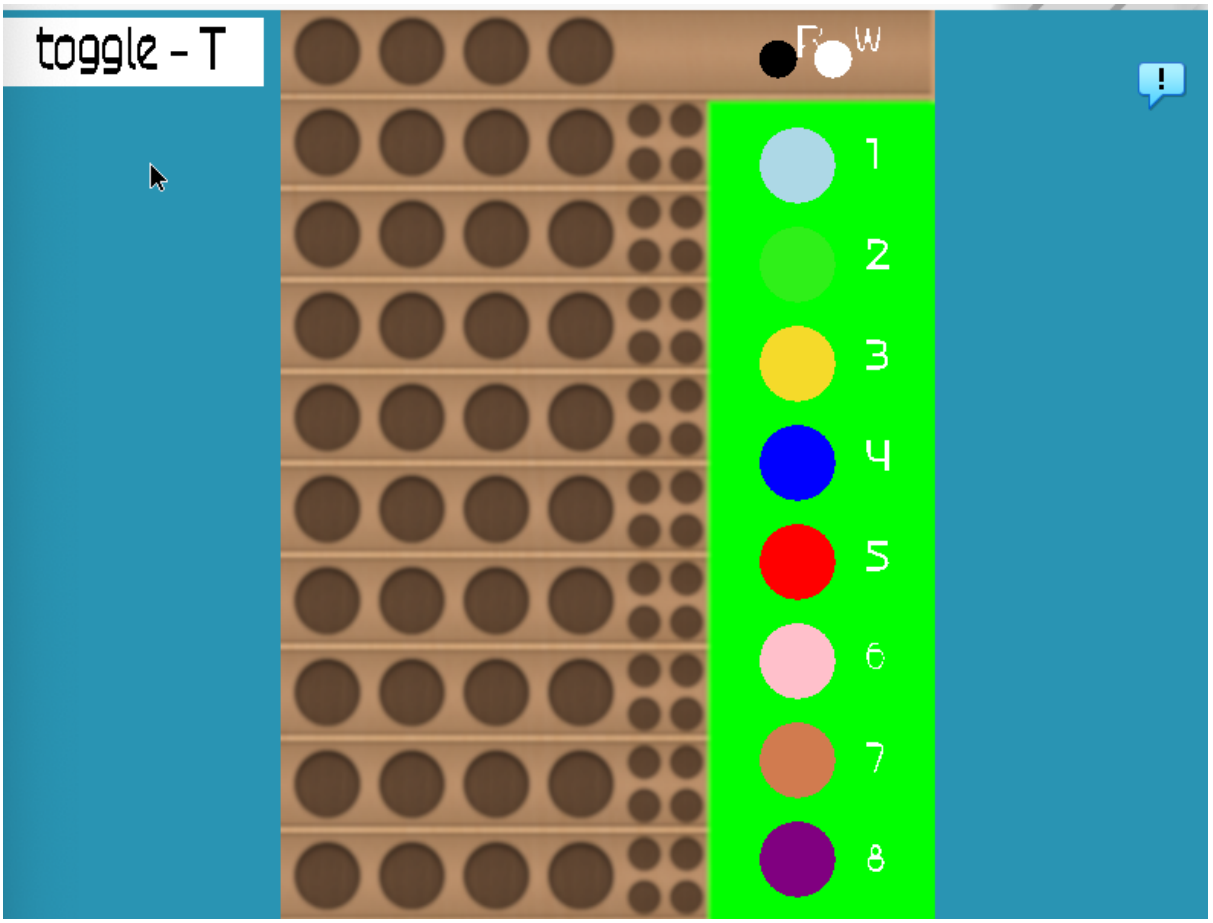


Fig.3 - Tabuleiro de jogo do jogador 2

Chave de jogo

A chave de jogo será introduzida pelo segundo jogador, na primeira linha do seu tabuleiro, clicando nas bolas e largando-as no sítio desejado.

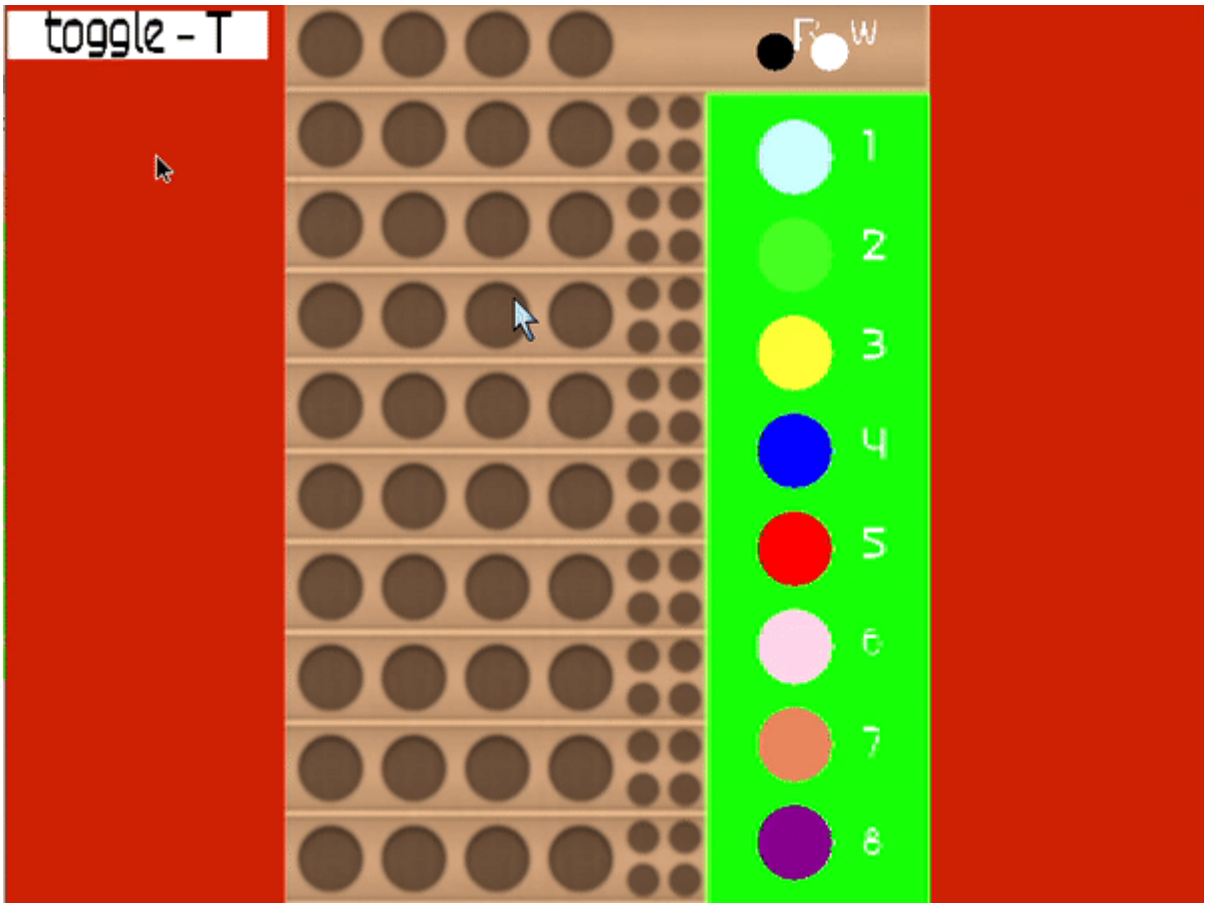


Fig. 4 - Introdução da chave no tabuleiro de jogo

Adivinhar a solução

O processo é semelhante à introdução da solução.

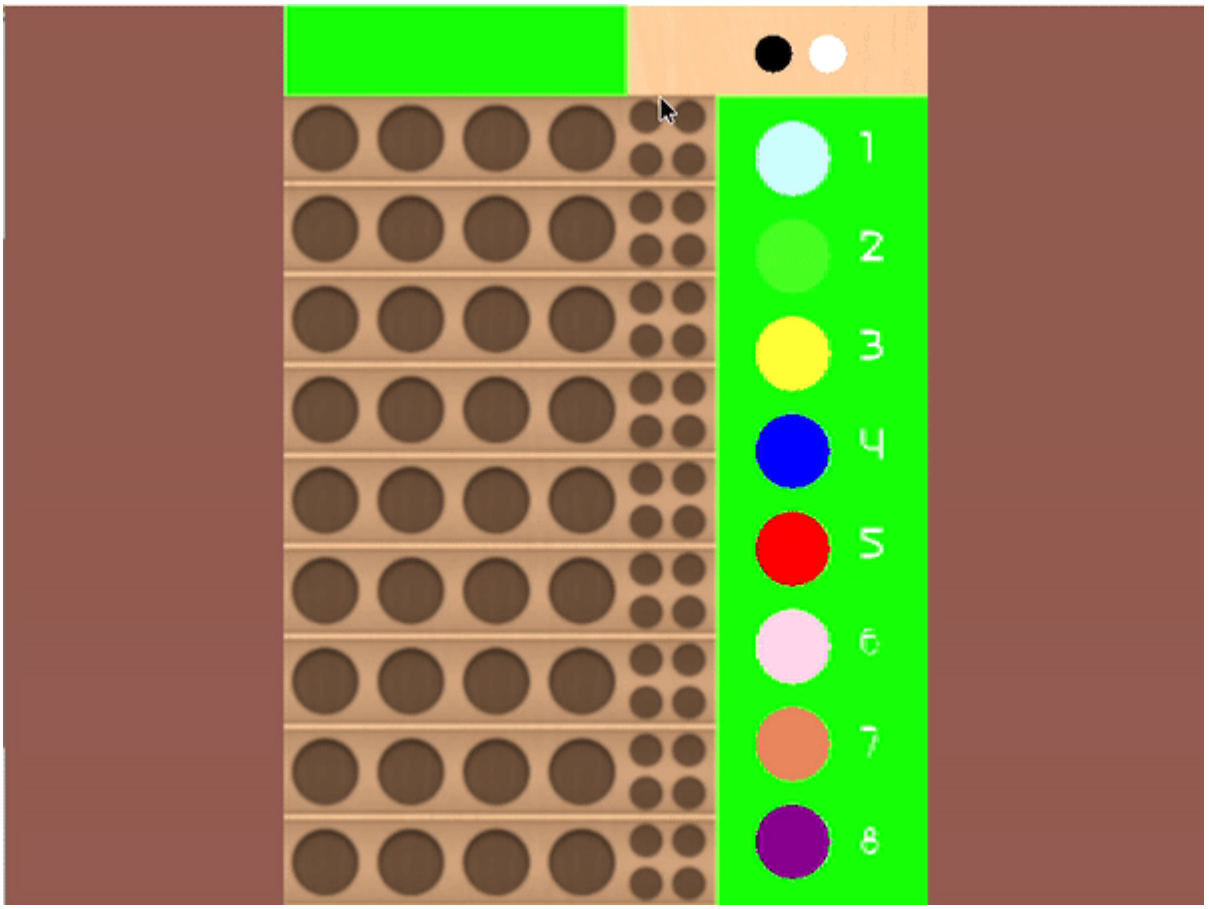


Fig. 5 - Tentativa de solução

Menus Finais

No caso do jogador 1 adivinhar o código secreto, será redirecionado para um menu em que o avisa de tal, bem como mostra a solução vencedora.

Fig. 6 - Menu vencedor

Menus Finais

No caso do jogador 1 não adivinhar o código, será redirecionado para um menu semelhante, onde a única diferença será uma mensagem de insucesso.

Fig.7 - Menu perdedor

Menus Finais

Caso o jogador 2 faça batota, não passando a chave certa (com as bolas pretas e brancas) para o jogador 1, ambos os jogadores serão notificados no final dessa ronda de jogadas.

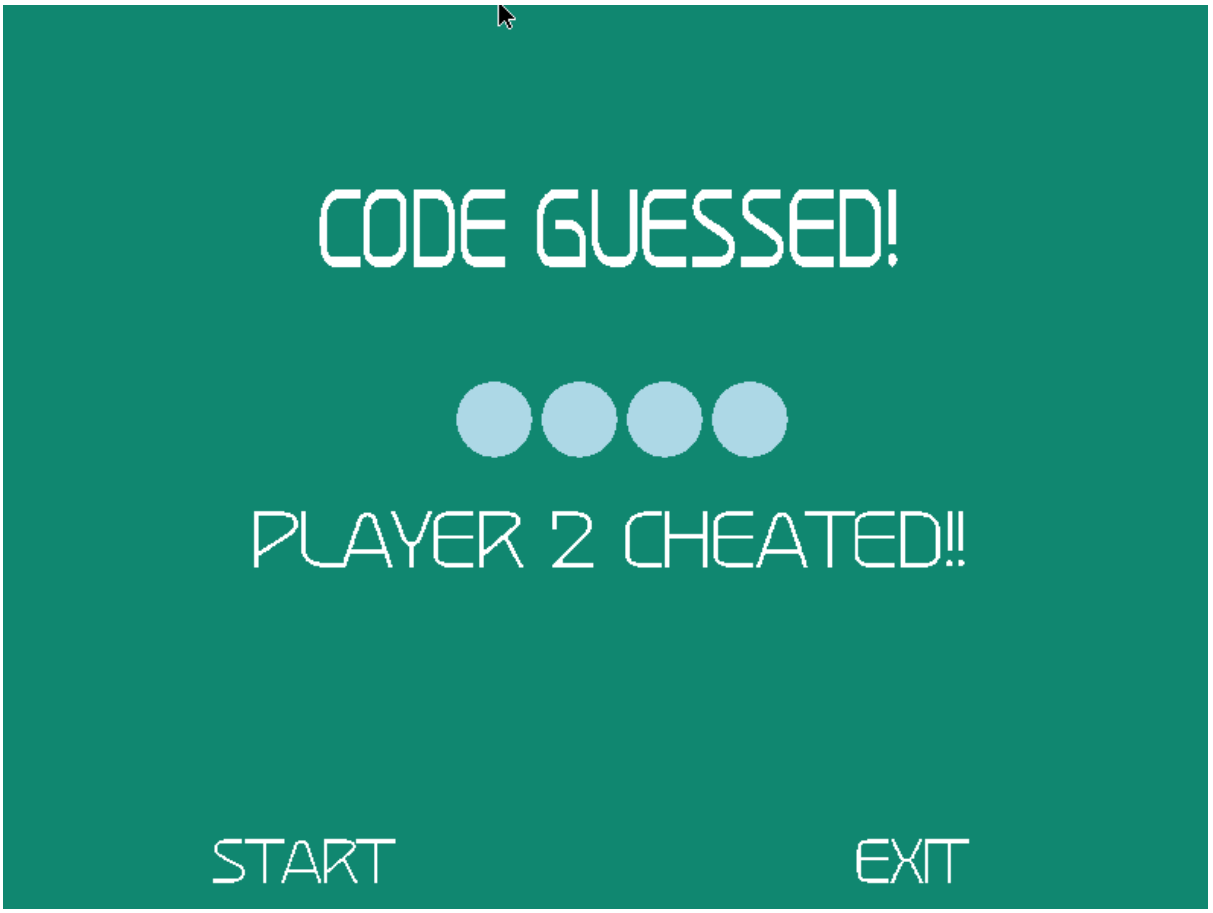


Fig.8 - Menu final no caso de batota

Estado do Projeto

Tabela de funcionalidades

Funcionalidades	Dispositivos	Estado de implementação
Pegar / Colocar / Retirar Bolas	Mouse, Teclado, Video Graphics	Completo
Comunicação entre os dois utilizadores	Serial Port e Timer	Completo
Navegar entre Menus	Mouse, Teclado, Video Graphics	Completo
Esconder código	Mouse, Teclado, Video Graphics	Completo

Tabela de dispositivos

Dispositivo	Descrição	Modo
Timer	Mantém o ritmo do jogo, principalmente no que toca à framerate	Interrupts
Teclado	Atalhos, confirmação de fim do turno	Interrupts
Rato	Movimento do cursor, manipulação de peças	Interrupts
GPU	Desenhar a interface do jogo	N/A
RTC	Definir a cor do	Polling

	fundo do ecrã	
Serial Port	Comunicar dados do jogo entre as duas VMs	Interrupts

Timer

O timer é usado para manter uma frequência de 60 Hz. O timer também é utilizado para efetuar certas operações a cada segundo [update_timer_state, model.c], como atualizar a fila ou ir buscar dados do RTC.

Teclado

O teclado é usado para o jogador confirmar o seu turno e para diversos atalhos de ações que são normalmente realizadas com o rato.

Rato

O rato é usado para atualizar a posição do cursor, para selecionar peças e clicar em botões (botão esquerdo), e para remover peças já colocadas (botão direito).

Estrutura de dados: MouseInfo, que contém as coordenadas x e y do cursor, a cor da bola atualmente pegada, e o estado dos botões esquerdo e direito

GPU

A placa gráfica é usada para desenhar a interface do jogo, principalmente através de sprites. O modo de vídeo usado foi o 0x115 (800x600, cor direta, 8:8:8 (RGB)).
Double buffering está implementado por cópia [copy_buffers, model.c], mas, além do double buffering, implementamos outra otimização que permite reduzir consideravelmente o número de

píxeis desenhado a cada frame ao redesenhar apenas os pedaços do ecrã em que se encontram elementos que não fazem parte do background [draw_partial_sprite_xpm, view.c].

Funções VBE usadas:
01h Return VBE Mode Information (através da LCF)
02h Set VBE Mode

RTC

O Real-Time Clock é usado para determinar a cor usada no fundo do ecrã, com uma cor para cada hora do dia.

Estrutura de dados: RTCInfo, contém informação sobre as horas (formato 0-23), minutos e segundos

Serial Port

O Serial Port é usado para que os movimentos de cada jogador sejam transmitidos ao outro. Interrupts são usados na transmissão (devido a ser possível ser gerada a interrupção de THR Empty quando não há dados a enviar, uma flag booleana é ativada para indicar que se pode transmitir [sp_ih, sp.c]) e na receção. Os FIFOs não foram usados, mas implementamos uma fila em software para a transmissão [queue.c] (detalhes na seção 4). Quanto aos dados em si, conseguimos condensar toda a informação de um movimento num byte [prepare_move_byte, sp.c] ao usar uma tabela indexada de cores. O byte é formado da seguinte forma:
-bits 0 a 3: índice da cor
-bit 4: se ativo, o bit é “especial” e não representa um movimento (ACK, NACK, fim de turno, fim de jogo, confirmação de número do jogador, restart do jogo)
-bit 5: se ativo, o movimento corresponde a uma remoção de peça, caso contrário a uma colocação
-bits 6 e 7: posição em que o movimento ocorreu (0 a 3)

Os dados não são transmitidos com uma frequência fixa, mas sim apenas quando acontece um movimento. Quando um byte é lido, é enviada uma resposta positiva (ACK) ou negativa (NACK) [write_sp_data/read_sp_data, sp.c] para saber se é preciso repetir a mensagem.

Organização do Código

Inicialmente seguimos o modelo MVC que aprendemos no semestre passado em na cadeira de LDTS. No entanto, não fomos muito diligentes ao seguir o modelo (principalmente ao colocar parte do código que deveria pertencer aos controladores no *Model Module*) acabando por reorganizar um pouco a estrutura.

Drivers

Timer - 5%

Idêntico ao do Lab2, contém funções para subscrever/cancelar subscrição das interrupções, definir a frequência e ler a configuração.

Teclado - 5%

Idêntico ao do Lab3, contém funções para subscrever/cancelar subscrição das interrupções, ler *scancodes* e *status* e escrever comandos.

Rato - 10%

Idêntico ao do Lab4, contém funções para subscrever/cancelar subscrição das interrupções, ler scancodes e status.

KBC - 2%

Idêntico ao dos Labs 3 e 4, contém funções para escrever comandos, ler *outputs* e *status*.

Vídeo - 10%

Semelhante ao Lab5, contém funções para pintar pixels, definir o *frame buffer* e desenhar linhas e retângulos.As funções relacionadas com XPM *loading* e desenho dos mesmos foram movidas para os *view* e *sprite modules*.

RTC - 1%

Semelhante ao que seria encontrado no Lab6. Constituído por funções para ler/escrever em qualquer um dos *registers*, obter informações sobre a configuração do RTC e preencher a estrutura de dados com os dados de tempo. Não possui funções relacionadas com interrupções, uma vez que as interrupções periódicas tinham um *upper bound* de 0.5 segundos, o que consideramos desnecessariamente pequeno, tendo em conta a forma como utilizamos o RTC. Este módulo contém também as funções usadas para manter um funcionamento correto do programa independentemente da configuração do RTC: [convert_output, rtc.c] e [convert_hours, rtc.c].

UART - 10%

Funções que gerenciam a serial port, como configurá-la (*bit rates*, habilitar interrupções no IER…) [sp_setup, sp.c],

subscrever/cancelar subscrição das interrupções, enviar dados, escrever dados e ler o *status*. Também contém a função que prepara *bytes* relacionados com as ações do jogador [prepare_move_byte, sp.c].

Lógica

Board Logic - 3%

Funções relacionadas às regras do jogo, que consistem em verificar se o jogador 1 conseguiu adivinhar o código criado pelo jogador 2 e se as dicas deste estão corretas.

Queue - 7%

Armazena a *queue* usada para enviar dados para a UART possibilitando dar *push* e *pop* a um *byte*, verificar se a *queue* está vazia e enviar o código para o jogador 1 quando o jogo termina.

Model - 25%

Este módulo contém grande parte das funcionalidades do jogo. Inclui as configurações dos diversos elementos tal como posição das bolas, sprites, estado da serial port do timer, do rato e do teclado... Possui ainda funcionalidades para colocar e remover bolas, terminar os turnos dos jogadores, seleção dos menus, entre outros. Também é neste módulo que está presente a função que determina o número do jogador no início do jogo [test_player_no, model.c].

View - 15%

Faz a ligação entre o as funcionalidades e o utilizador através do display dos componentes do programa. Inclui funções para desenhar xpms, menus e bolas (diversas funções cujo nome inicia por “draw”), e também para “limpar” sprites (isto é, desenhar o background por trás das suas posições, em diversas funções cujo nome inicia por “clean”).

Detalhes de implementação

UART

Como conseguimos condensar tudo o que queríamos transmitir em pacotes de um byte, usamos uma implementação relativamente simples da UART, usando apenas as suas features mais simples com interrupções. A maioria do código mais importante relativo à transmissão de dados encontra-se no módulo da queue.

RTC

Com o uso que o RTC tem no nosso projeto, na verdade só precisaríamos de ir buscar os dados relativos às horas, mas os dados são obtidos a cada segundo (se o RTC estiver a atualizar, ignora-se a tentativa e os dados permanecem iguais durante outro segundo). Tivemos o cuidado de preparar conversões de BCD para binário e de formato de 12 horas para 24.

Queue

Implementamos uma fila em software para poder rapidamente preparar múltiplos bytes para transmissão e para evitar erros de sobreposição de caracteres. A fila consiste num máximo de 16 bytes, com um buffer adicional de apenas um byte. Para evitar usos desnecessários de memcpy para deslocar os elementos da fila um byte para a esquerda a cada operação de “pop” [pop, queue.c], tornamos a cabeça da fila dinâmica, deslocando-se um byte para a esquerda logo após um pop. Quando a UART está pronta para transmitir um byte, a fila sofre um pop (se tiver bytes), o topo da fila é colocado no buffer de um byte e é enviado para a UART. Se a outra VM responder com um NACK, a fila fica pronta para transmitir mais dados e o byte usado antes é considerado como descartado. No entanto, se a resposta for NACK, a queue mantém o byte no buffer de um byte e tenta enviar outra vez o byte para a UART até haver uma resposta satisfatória [retry, queue.c].

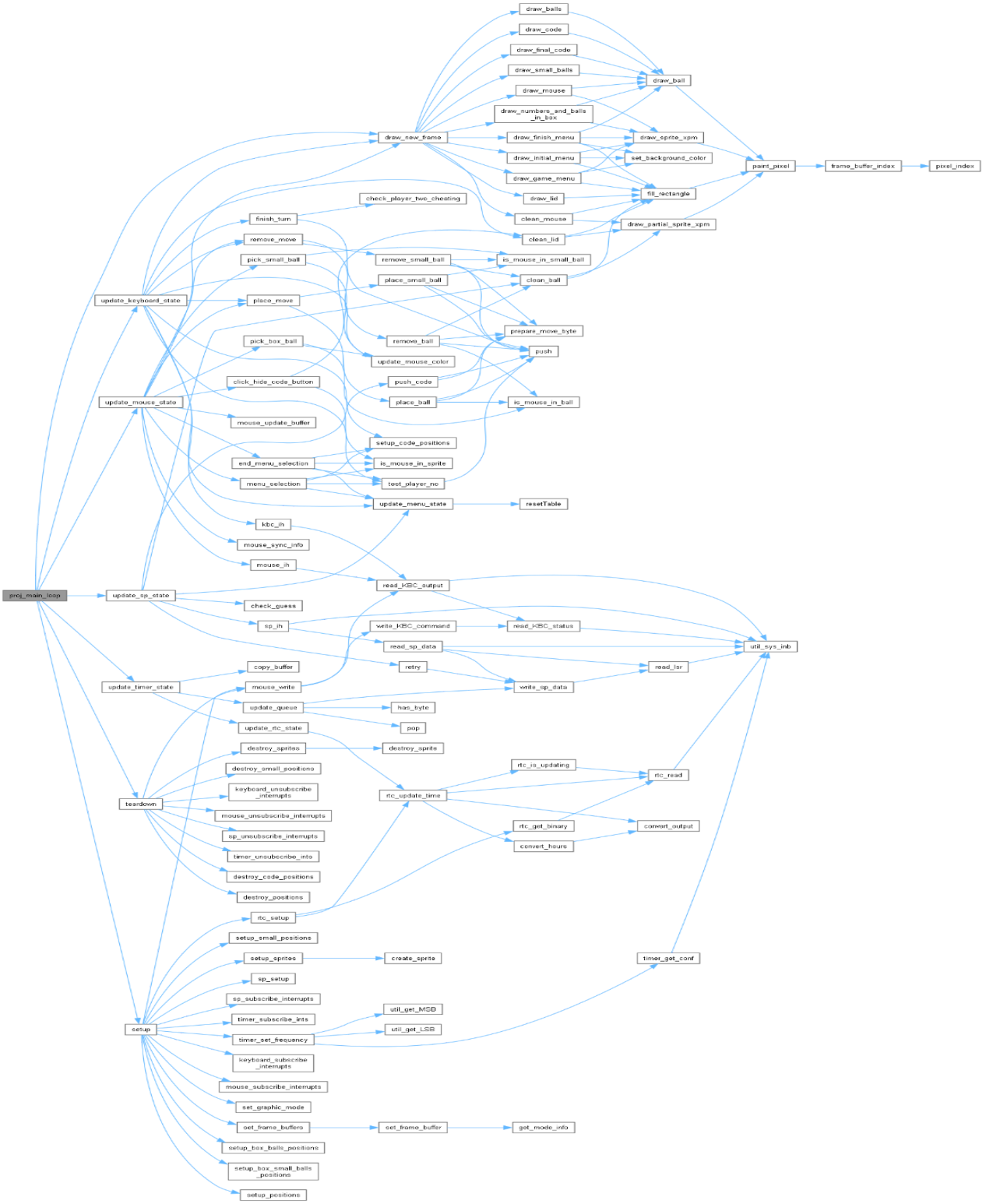
Background

Para evitar redesenhar a integralidade do frame buffer a cada refresh, criamos um sistema de background, usado no módulo view. Quando nos deslocamos a um novo menu, uma nova cor de background é designada, e um array de Sprites é preenchido com as camadas do fundo do novo menu. Na primeira frame, tem de ser tudo redesenhado, como é óbvio. No entanto, depois disso, apenas as secções do ecrã ocupadas por outras sprites são desenhadas [draw_partial_sprite_spm, view.c].

Troca de turnos entre os utilizadores

De forma a criar um flow de jogo saudável e a troca de turnos entre jogadores/virtual boxes, após cada jogador terminar a sua jogada, deve pressionar a tecla “ENTER” de forma a dar a vez ao seu oponente [finish_turn, model.c].

Function call Graph



Conclusões

Conseguimos implementar todas as features que planeamos inicialmente com tempo para adicionar mais uns retoques, como a capacidade do jogador 2 esconder o código no seu ecrã. No entanto, duas features que seriam adicionadas ao jogo se continuássemos a trabalhar nele seria a capacidade de trocar o papel de cada jogador entre partidas e a capacidade de jogar sozinho (a chave seria gerada aleatoriamente). Um aspeto com o qual ficamos satisfeitos foi a implementação da UART, que, embora meio simplista quanto ao uso das suas capacidades, funciona sem problemas aparentes, mesmo numa máquina mais lenta (um dos membros do grupo tem um computador da Apple).