U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Paint with Friends

Projeto realizado no âmbito da cadeira de Laboratório de Computadores

Turma 9 - Grupo 1

Trabalho realizado por:

- Carlos Daniel Lopes Rebelo up202108885
- Diogo Tomás Valente Fernandes up202108752
- Hélder Gabriel Silva Costa up202108719
- Jaime Francisco Rodrigues Fonseca up202108789

Índice

Introdução

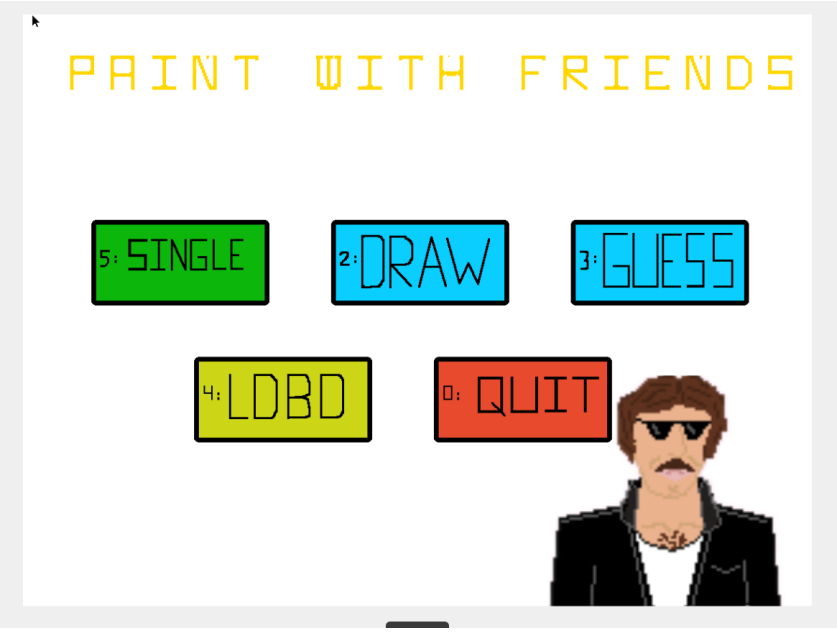
O nosso projeto é inspirado no jogo de tabuleiro “Pictionary”. A premissa base do jogo é relativamente simples; um dos jogadores estará a desenhar uma palavra que lhe será fornecida aleatoriamente e o outro jogador terá de adivinhar aquilo que está a ser desenhado. O objetivo é tentar adivinhar o mais rapidamente possível pois isso será consequentemente relacionado com a pontuação obtida.

Instruções de utilização

Menu Inicial

Ao iniciar o programa, será apresentado um menu inicial, com o título do nosso projeto, e com botões diferentes que permitirão ao utilizador navegar pelos diferentes modos de jogo. O utilizador poderá selecionar para onde deseja ir de duas maneiras diferentes: utilizando o rato e simplesmente clicar em cima do botão em questão, ou utilizando alguns dos números do teclado que estão representados nos botões. As funcionalidades dos botões são:

- **Singleplayer -** Utilizador é direcionado para um modo de jogo onde lhe é permitido tanto desenhar como adivinhar ao mesmo tempo. A palavra que é suposto desenhar aparece durante 5 segundos e após esse tempo passar, é permitido começar a escrever tentativas de resposta.
- **Coop -** Modo de jogo com serial port onde a um jogador lhe é permitido desenhar e não escrever e vice-versa ao outro jogador.
- **Leaderboard -** Permite ver as melhores pontuações de maneira decrescente (quanto mais rápido se adivinhar a palavra mais pontos se ganha). Está também registrado quando foi feita a pontuação em tempo real.
- **Quit -** Permite sair do jogo e voltar ao modo de texto do MINIX.



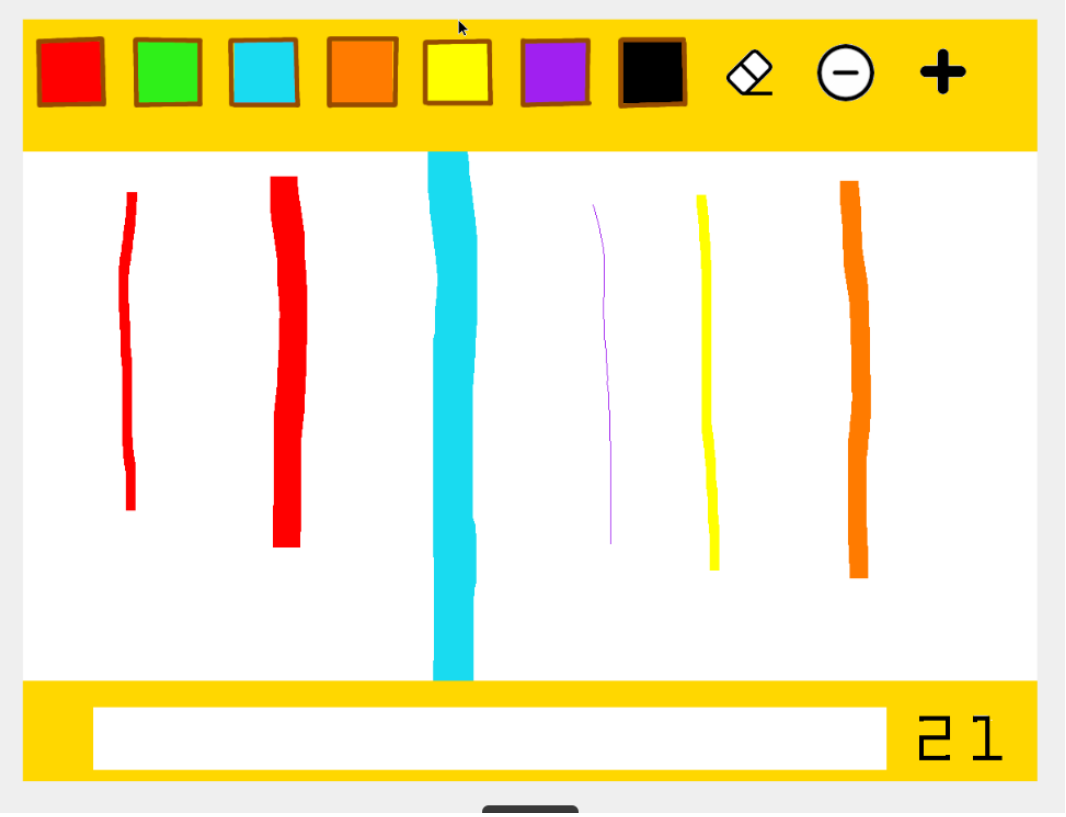
Menu de jogo

No menu de jogo, como explicado anteriormente, dependendo do modo de jogo, é possível desenhar, escrever, ou ambos.

Desenhar

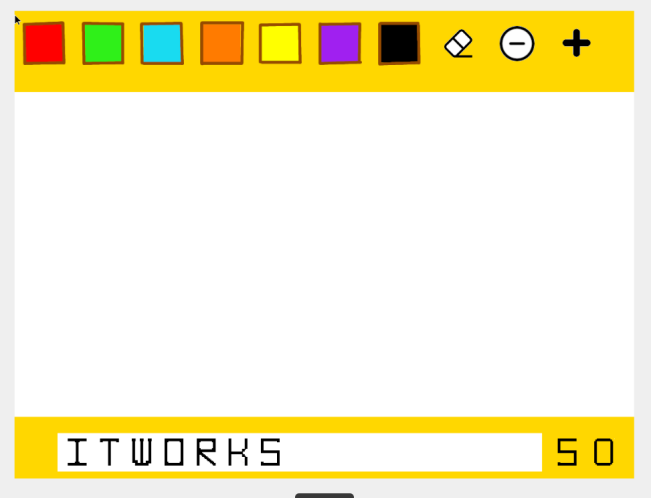
Relativamente à funcionalidade de desenhar algumas das features que implementámos foram:

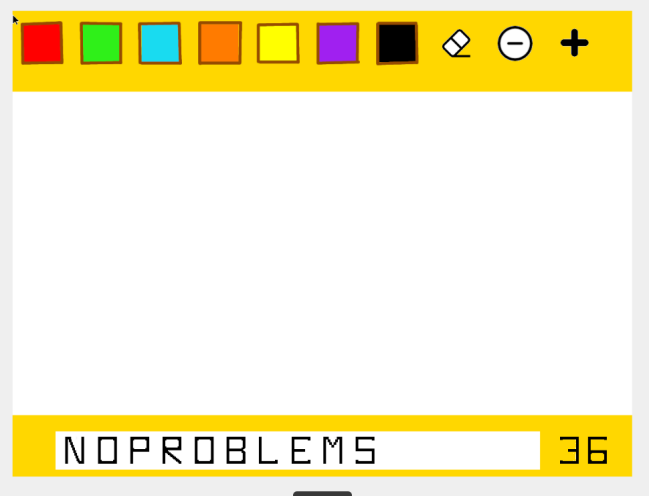
- Desenhar com cores diferentes
- Desenhar com diferentes espessuras (espessura mínima de 1 pixel por cada vez que se pressionar o left-button do rato)
- Apagar aquilo que foi desenhado



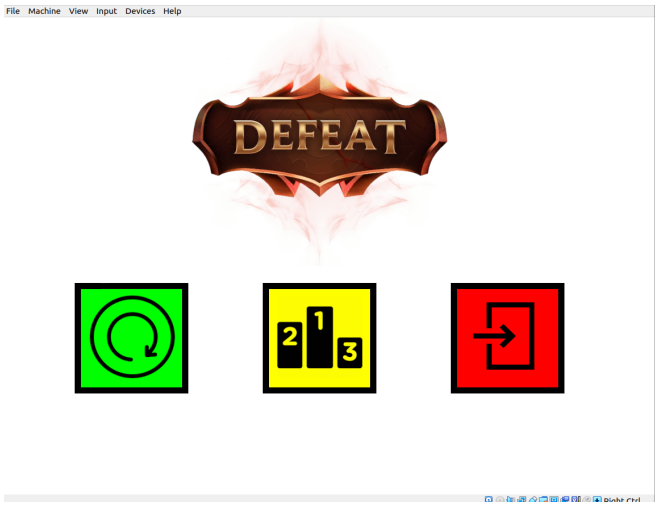
Adivinhar

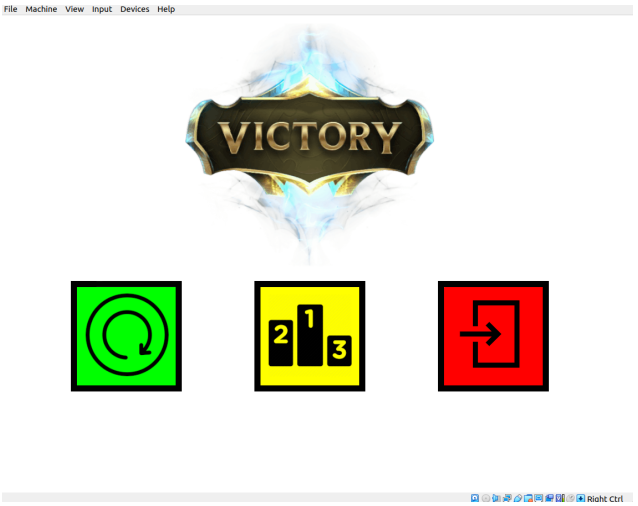
Para o jogador responsável por adivinhar aquilo que está a ser desenhado no ecrã , terá de o fazer usando o teclado, estando a palavra que está a escrever representada numa barra na parte inferior do ecrã. Estará também, um cronómetro que estará a decrementar o seu valor a cada segundo.





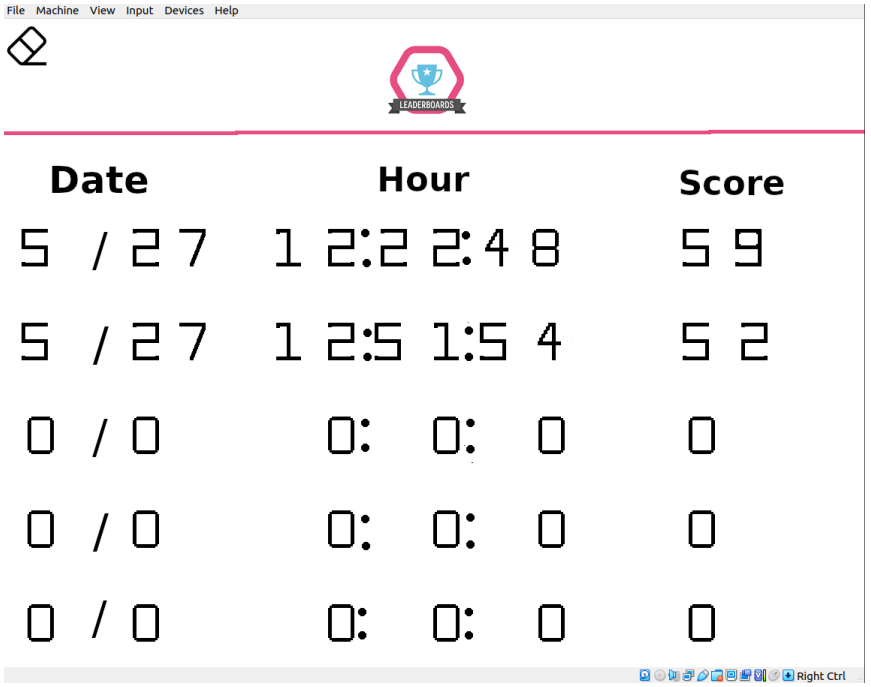
Para verificar se a sua tentativa de resposta está correta ou não, o jogador pressiona na tecla “Enter”. Existem então 2 cenários possíveis: a resposta estar correta e o jogador ser direcionado para uma outra página que lhe dirá que conseguiu ganhar o jogo e onde conseguirá voltar a repetir o jogo, ir para a leaderboard etc. Se a resposta estiver incorreta nada acontece. Caso o tempo acabe e não tenha conseguido adivinhar a palavra em questão é direcionado para a mesma página que iria caso tivesse ganho, com, obviamente, a diferença de aparecer a informação de que perdeu o jogo.





Menu da leaderboard

No menu da leaderboard, é possível ver as melhores pontuações (que são guardadas mesmo se fecharmos o jogo) e também é possível "limpar" os valores da leaderboard se clicarmos na borracha no canto superior esquerdo do ecrã.



Estado do Projeto

Tabela de funcionalidades

Funcionalidade	Dispositivos	Estado de Implementação
Desenhar	Rato e placa de vídeo	Completo
Mudar a cor / espessura	Rato e placa de vídeo	Completo
Apagar	Rato e placa de vídeo	Completo
Mostrar o cronómetro de	Timer e placa de vídeo	Completo

um jogo		
Mostrar os melhores resultados na leaderboard	Rtc, placa de vídeo e timer	Completo
Mostrar a palavra para desenhar	Placa de vídeo	Completo
Mostrar a tentativa de palavra correta	Teclado e placa de vídeo	Completo
Navegar entre os diferentes menus	Teclado, rato e placa de vídeo	Completo
Comunicar entre diferentes computadores	Serial Port, Timer(?)	Incompleto(?)

Tabela de dispositivos

Dispositivos	Funcionalidades	Interrupções
Timer	Controlar a frame rate, decrementar o cronómetro do jogo	Sim
Teclado	Navegar entre os menus, escrever a tentativa de palavra correta	Sim
Rato	Desenhar, mudar as especificações de desenho (cor , espessura, apagar etc), navegação entre menus	Sim
Placa de vídeo	Desenhar toda a interface do jogo, incluindo xpms, aquilo que está a ser desenhado, tempo de jogo etc	Não(?)

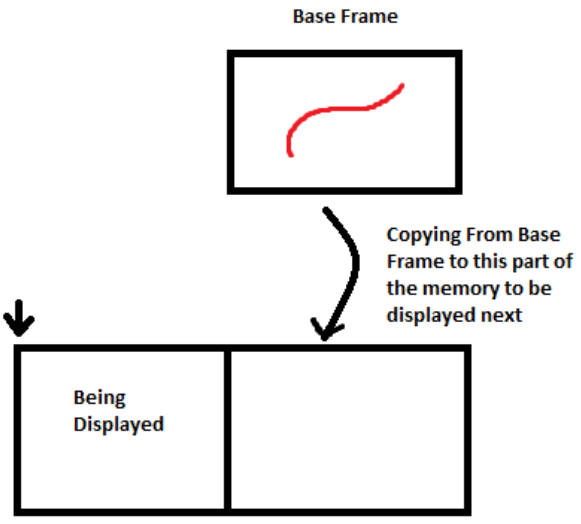
Dispositivos

Placa de vídeo

O modo de vídeo utilizado no projeto foi o modo 0x14c. A resolução deste modo é de 1152 pixels sobre o eixo dos x por 864 pixels sobre o eixo dos y. A cor de cada pixel neste modo está configurado em modo direto, com 32 bits por pixel - 8 bits por cada componente RGB mais 8 bits para a opacidade.

Para aumentar a fluidez ao desenhar, adotamos a estratégia de page flipping bem como a existência de um buffer auxiliar, base frame foi o nome dado no projeto.

No buffer auxiliar está registado apenas aquilo que foi ou está a ser desenhado pelo jogador. Esta estratégia foi utilizada dada a natureza do nosso jogo, para assim, não se perder informação relativamente aos pixels já pintados, quando passarmos o rato por cima deles, por exemplo. Aquilo que é feito é uma cópia da informação que o base frame tem para a page que não está a ser displayed no momento, bem como uma mudança da page que está a ser exibida a cada interrupção.



Para limpar o conteúdo deste base frame, algo necessário entre ocorrências do nosso menu de desenho, por exemplo, utilizamos uma função chamada reset_frame. Na classe do base_frame há também funções para desenhar pixels, que desenham especificamente neste

buffer, enquanto que há uma função similar que desenha diretamente no buffer da placa de vídeo. Por questões de organização fizemos esta divisão.

Utilizamos sprites para criar alguns dos elementos visuais do nosso programa. Para o efeito criámos uma classe chamada Sprite. Ao iniciarmos o nosso programa executamos uma função chamada setup_sprites() que irá criar todos os objetos sprites que serão necessários a partir dos ficheiros xpm que temos guardados na diretoria *model/xpm*. Pelo ponto de vista das boas práticas ao pararmos a execução do programa, destruímos os objetos criados. A configuração e as funcionalidades relativas à placa de vídeo estão no ficheiro *graphics.c*.

Teclado

No contexto do nosso projeto, o teclado foi utilizado com dois propósitos:

- Permite escrever a palavra que o utilizador acha ser correta para ganhar o jogo.
- Permite verificar se a tentativa de resposta está correta.
- Permite ao utilizador mudar entre os diferentes menus do projeto:
  - “1” - Menu principal
  - “0” - Sair do jogo
  - “5” - Modo "single player" que permite desenhar e adivinhar a palavra ao mesmo tempo
  - “ENTER” - Verificar se a sua tentativa de resposta está correta

A configuração e as funcionalidades relativas ao teclado estão no ficheiro *keyboard.c*.

Rato

O mouse é utilizado ao longo de todo o projeto permitindo fazer praticamente tudo. Permite-nos navegar entre os diferentes menus ,

desenhar, mudar as características da linha que estamos a desenhar, sair do jogo, etc.

Consequentemente, precisamos não só de manter um registo relativamente à posição do mesmo, como também estar atento relativamente aos botões que podem ou não, ter sido pressionados. Isto porque, para fazer uma linha contínua, por exemplo, é necessário manter o botão esquerdo do mouse pressionado. Para além disso, clicar no botão direito do rato, no modo de desenho permite-nos limpar tudo aquilo que possa ter sido desenhado até aquele momento.

De cada vez que a posição do rato é alterada, é necessário desenhar o xpm do rato também numa posição diferente. Para o fazer temos duas variáveis (int) a que chamamos x e y. Estas variáveis são então responsáveis por manterem o registo da posição do rato relativamente ao ecrã de cada vez que a sua posição é alterada. Manter um registo destas variáveis permite-nos também, logicamente, identificar qual dos botões apresentados foi "pressionado" pelo utilizador e mudar o comportamento do nosso programa de acordo com aquilo que aquele botão seria suposto fazer. A configuração e as funcionalidades relativas ao teclado estão no ficheiro *mouse.c*.

Timer

O timer (mais especificamente o timer 0) é utilizado com três propósitos diferentes no projeto, embora dois deles sejam bastante semelhantes. Um deles é controlar a frame rate da placa de vídeo, que no nosso caso em concreto, foi definido para 30 interrupções por segundo (30 fps).

Para além disso, existem duas variáveis chamadas game_counter e delay. A variável game_counter mantém um registo e limita o tempo de cada jogo (definido a 60 segundos para cada jogo). A variável delay é uma variável que é usada no modo de jogo onde é possível tanto desenhar como adivinhar a palavra, que está inicialmente definida para 5 segundos. Esta variável é, à semelhança da game_counter, decrementada a cada 30 interrupções (a cada segundo) e o objetivo é que durante os primeiros 5 segundos, neste modo de jogo, seja demonstrado a palavra que corresponde à "solução" do jogo, de outra maneira, a

palavra a ser desenhada por um utilizador e adivinhada por outro. Após esses 5 segundos terminarem, a solução desaparece e é agora permitido escrever para tentar adivinhar e ganhar o jogo.

A configuração e as funcionalidades relativas ao timer estão no ficheiro *timer.c*. As duas variáveis explicadas, encontram-se no ficheiro *model.c*.

RTC

O rtc no nosso projeto foi utilizado com o propósito de manter um registo do data time real em que uma determinada pontuação foi conseguida para ser registada na nossa leaderboard.

Para manter um registo do tempo atual real, temos uma variável chamada curr_time, presente no ficheiro *model.c*. Esta variável é do tipo struct, criada por nós e chamada real_time que possui informação como os segundos, minutos etc. Por motivos de performance, esta variável apenas é utilizada a cada segundo que passar, visto que dada a natureza de um dispositivo como o rtc este estaria constantemente a ser atualizado, algo que sobrecarregaria o nosso projeto, para além de que, apenas nos interessa mudar a nossa variável a cada segundo, menos do que isso é, no nosso contexto, desnecessário.

Para este efeito, foi criada uma struct chamada leaderboardValue que irá conter alguma da informação do rtc, nomeadamente, o dia, o mês, a hora, os minutos e os segundos que serão atualizados de acordo com a informação no momento da nossa variável curr_time. Para além desta informação tem um outro atributo denominado score que irá registar quanto tempo faltava para acabar o jogo (quanto mais tempo faltar melhor).

Sempre que um jogo acaba com uma vitória, por parte do utilizador que estava a tentar adivinhar, é criado um objeto “newValue” do tipo da struct já explicada. A organização destes valores presentes na leaderboard, não implica a utilização do rtc em si, utilizando funções por nós criadas que serão explicadas em maior detalhe mais à frente.

Organização do código e estrutura

Neste projeto decidimos usar a arquitetura MVC que permite que o código seja separado perante a função que vai representar (model, view ou controller) e permite também uma melhor manutenção e reutilização de código.

Placa de vídeo - 12%

Este módulo (*graphics.c*) contém as funções do lab5 desenvolvidas nas aulas práticas que foram realizadas ao longo do semestre. Mantivemos aquelas que seriam as funções que fariam sentido manter no contexto do nosso projeto. Para além disso, adicionámos algumas funções que, no contexto da arquitetura de um MVC considerámos fazer sentido estarem colocadas aqui, nomeadamente as funções relativas ao processo do page flipping (vg_flip_frame()) ou à cópia do base frame para a página do buffer da placa de vídeo seguindo a lógica já previamente explicada.

As principais estruturas de dados presentes são a video_mem (endereço de memória do buffer da placa de vídeo) ou o page_index que é constantemente alterado e que tal como o nome indica indica para qual page, no contexto do page flipping, temos de copiar o conteúdo do base frame;

Teclado - 10%

Este módulo (*keyboard.c*) contém as funções do lab3 desenvolvidas nas aulas práticas que foram realizadas ao longo do semestre, que será responsável principalmente por lidar com as interrupções geradas pelo controlador do keyboard sempre que uma tecla do nosso teclado é pressionada por um utilizador.

A acrescentar a estas funções, existe uma outra função por nós implementada, durante o projeto chamada read_letter(). Esta função é utilizada quando um dos jogadores está a tentar adivinhar a palavra que corresponde ao que está a ser desenhado no ecrã. O objetivo desta função é traduzir o scancode da tecla para um offset. Offset esse que será

utilizado para sabermos dentro do xpm das letras, quantos pixels temos de nos deslocar para garantir que no ecrã, estamos a desenhar a letra correspondente à tecla pressionada. Esta função recebe como argumentos:

- scancode -> Corresponde ao scancode da tecla que foi pressionada.
- word_guess -> Neste array temos guardados os offsets de todas as letras que foram pressionadas (e não foram apagadas), em específico, da palavra que o nosso jogador está a escrever para tentar adivinhar o que está a ser desenhado. Funcionam no nosso projeto como uma espécie de "id" de cada letra. Fazemos assim, uma tradução "direta" de um scancode para aquilo que nos é realmente útil ao longo do projeto.
- number_letter -> Quantas letras tem a tentativa de resposta (pode ser incrementado ou decrementado se adicionarmos ou retirarmos uma letra)

Para além do scancode, não há estruturas de dados específicas deste módulo importantes de serem realçadas, apenas estas que são passadas como argumento da função.

Rato - 12%

Este módulo (*mouse.c*) contém as funções do lab 4 desenvolvidas nas aulas práticas que foram realizadas ao longo do semestre.

Timer - 8%

Este módulo (*timer.c*) contém as funções do lab 2 desenvolvidas nas aulas práticas que foram realizadas ao longo do semestre, mantivemos funções que achamos relevantes para este projeto não tendo sido acrescentadas quaisquer funcionalidades para além das já desenvolvidas.

Rtc - 5%

Este módulo (*rtc.c*) contém todas as funções necessárias ao funcionamento do rtc no nosso projeto. Nomeadamente as funções de subscribe e unsubscribe (rtc_subscribe_int() e rtc_unsubscribe_int()) assim como as funções associadas à leitura e inicialização do rtc em si.

Para além disso, existe a estrutura de dados criada por nós, já referida anteriormente, real_time que está associada à variável curr_time, esta estrutura de dados guarda toda a informação sofre o momento em que é utilizada, desde segundos até ao ano que serão depois utilizados no âmbito de manter a leader board atualizada.

Base Frame Module - 8%

Este módulo contém a informação relativa ao buffer auxiliar que irá guardar toda a informação relativa àquilo que o jogador desenha, bem como algumas funções auxiliares como a draw_bresenham_line() que irá ser explicada em maior detalhe mais à frente.

Queue Module - 5 %

Classe auxiliar criada que nos permite utilizar e organizar a informação nela contida seguida da lógica FIFO (first in first out).

Sprites Module - 4%

Este módulo cria os elementos visuais a partir dos ficheiros xpm que criámos. A implementação foi baseada nos powerpoints do professor João Cardoso.

Model module - 15 %

Este módulo contém praticamente toda a informação necessária para o funcionamento do programa. Desde variáveis como a palavra que

um dos jogadores está a escrever como tentativa de resposta, como a palavra que é a solução do jogo em específico, aos sprites utilizados ao longo de todo o projeto etc. É também neste módulo que os states do nosso programa são alterados tendo em conta o input dado.

Existem 3 tipos de states diferentes:

- SystemState (Running / Exit) - Se o programa está a executar ou parou
- MenuState (Start / Game / Leaderboard / End) - Representa cada um dos menus de jogo
- GameState (Draw / Guess / Draw_Guess) - Dentro do menu de jogo existem 2 modos de jogo diferentes (o modo com a serial port implementada onde um jogador desenha e o outro adivinha ou o modo sem serial port onde é possível fazer os dois).

View Module - 10%

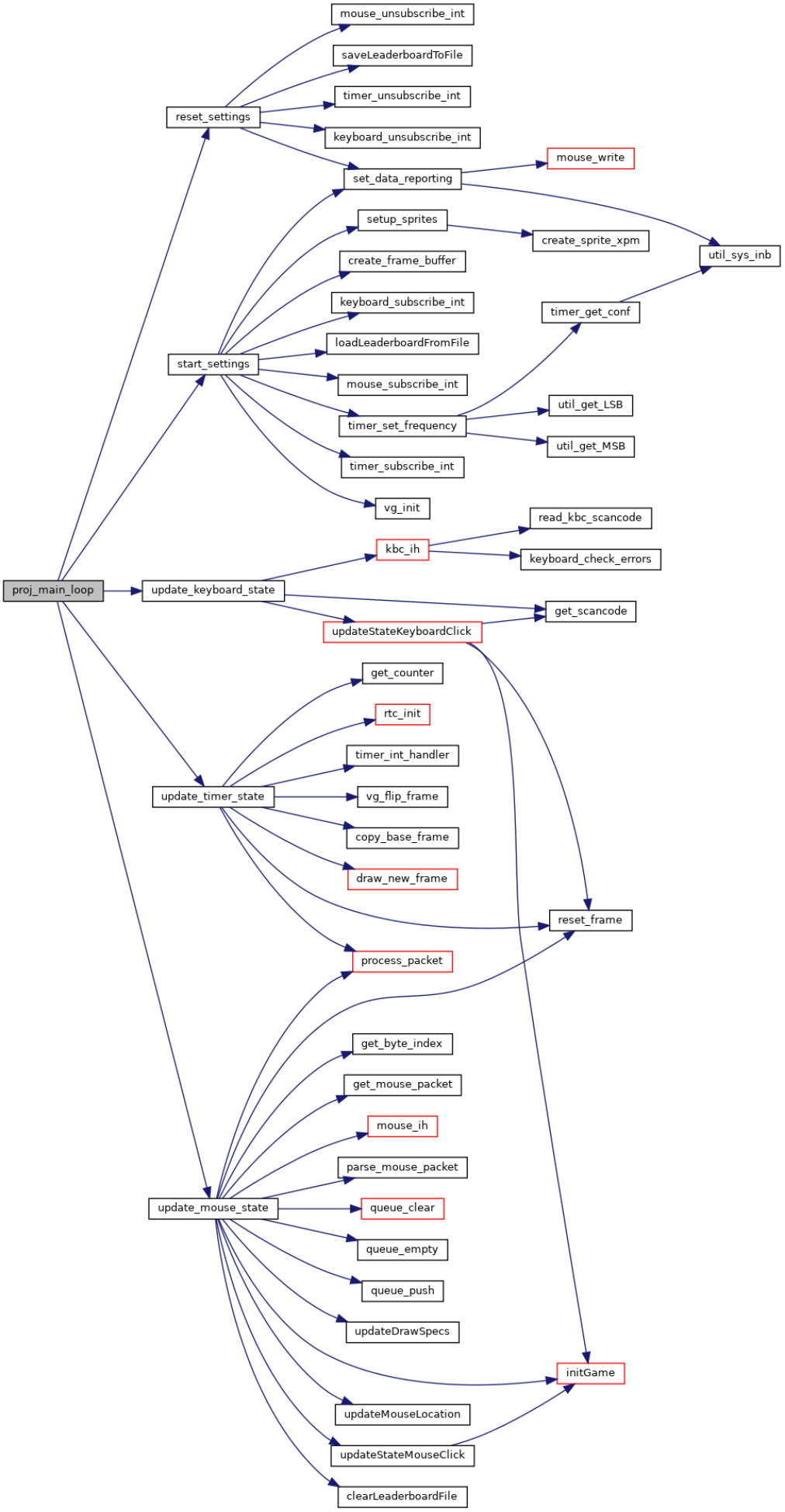
Este módulo é responsável por dar display de toda a interface gráfica do jogo, desde as sprites criadas àquilo que o nosso utilizador está a desenhar, tudo passará por este módulo para ser desenhado. Seguindo a lógica de uma arquitetura MVC, utiliza a informação armazenada no model para dar display ao utilizador.

Proj Module - 10%

Contém o loop principal, que irá chamar as funções que mudam o state do programa de acordo com a interrupção que recebam. Tem uma função start_settings que irá iniciar tudo aquilo que será necessário ao longo do projeto, desde ativar interrupções dos dispositivos,a criar as sprites etc. Temos também a função reset_settings que faz o processo oposto bem como prepara o regresso ao modo texto do MINIX.

Utils Module - 1%

Contém funções auxiliares, utilizadas pela maior parte dos dispositivos ao longo de todo o projeto. Funções como ler de uma port, transformar BCD em binário etc.


proj_main_loop
reset_settings
mouse_unsubscribe_int
saveLeaderboardToFile
timer_unsubscribe_int
keyboard_unsubscribe_int
set_data_reporting
mouse_write
start_settings
setup_sprites
create_sprite_xpm
create_frame_buffer
keyboard_subscribe_int
loadLeaderboardFromFile
mouse_subscribe_int
timer_set_frequency
timer_subscribe_int
vg_init
timer_get_conf
util_sys_inb
util_get_LSB
util_get_MSB
update_keyboard_state
kbc_ih
read_kbc_scancode
keyboard_check_errors
get_scancode
updateStateKeyboardClick
update_timer_state
get_counter
rtc_init
timer_int_handler
vg_flip_frame
copy_base_frame
draw_new_frame
reset_frame
process_packet
update_mouse_state
get_byte_index
get_mouse_packet
mouse_ih
parse_mouse_packet
queue_clear
queue_empty
queue_push
updateDrawSpecs
initGame
updateMouseLocation
updateStateMouseClick
clearLeaderboardFile

Detalhes da implementação

Utilização de uma queue para as posições do rato ao desenhar

Um dos detalhes da nossa implementação, foi a utilização de uma queue com o objetivo de traçar as linhas sem a presença de intervalos (“gaps”) caso se movimente o rato demasiado depressa. A cada interrupção do mouse atualizamos a posição do cursor , no entanto, a linha não está a ser desenhada à velocidade das interrupções do rato, criando os tais “gaps”.

Assim, a cada interrupt do rato, caso o botão esquerdo do rato esteja pressionado (condição necessária para desenhar) damos push da posição do rato para uma queue. Agora, a cada interrupção do timer, caso a queue tenha alguma coisa, vamos desenhar uma porção dessas posições. A porção que utilizámos foi 10 visto que foi a que considerámos ótima, visto que o computador não fica nem muito lento nem há atrasos significativos a desenhar (por estar a desenhar poucas vezes). A questão agora passa a ser, como devemos processar estes pacotes enviados para a queue. Existem 3 situações:

- A queue encontra-se vazia, logicamente não desenhamos nada como já foi mencionado.
- A queue tem dois ou mais elementos. Desenhamos 1 ponto até ao outro utilizando o algoritmo de bresenham, presente no nosso projeto na função draw_bresenham_line na classe base frame. O que esta função faz é, ao invés de desenhar constantemente círculos, correndo inevitavelmente o risco de estar a pintar por cima de algo já pintado, desenha logo a região de um ponto ao outro ponto, sendo mais eficiente por não estar a pintar por cima, bem como prevenindo a existência dos intervalos na linha. Damos pop apenas ao primeiro elemento da queue.
- A queue tem apenas um elemento. Desenhamos o ponto e não damos pop do primeiro elemento da queue. A diferença entre dar pop com dois ou mais elementos e não dar pop quando só houver um é o que permite manter sempre o

ponto anterior no topo da queue e que depois consigamos ligar esse ponto a outro, tornando a linha ininterrupta.

Conclusões

O jogo proposto, dado a sua natureza, requereu da nossa parte não só um domínio dos dispositivos abordados ao longo do semestre bem como, irmos mais além, abordando outros dispositivos como o rtc, que se revelou relativamente simples, e também o serial port.

As maiores dificuldades que enfrentámos neste projeto foram a implementação do page-flipping em conjunto com a existência do buffer auxiliar e a implementação da serial port, dado que foi um dispositivo que não foi abordado em detalhe nas aulas laboratoriais.

A única feature que estava inicialmente planeada mas que não foi executada, foi a transferência daquilo que está a ser desenhado de um computador para o outro via serial-port. No entanto, implementámos, por exemplo, o algoritmo de Bresenham, algo que não tínhamos planeado.

É opinião unânime no grupo que a realização deste projeto permitiu-nos não só consolidar aquilo que foi lecionado como aumentar o nosso aprendizado em relação a esta unidade curricular.

Consideramos, tendo em conta o âmbito geral, ter cumprido com o planeamento do projeto, resultando na apresentação de um produto final satisfatório e de qualidade.