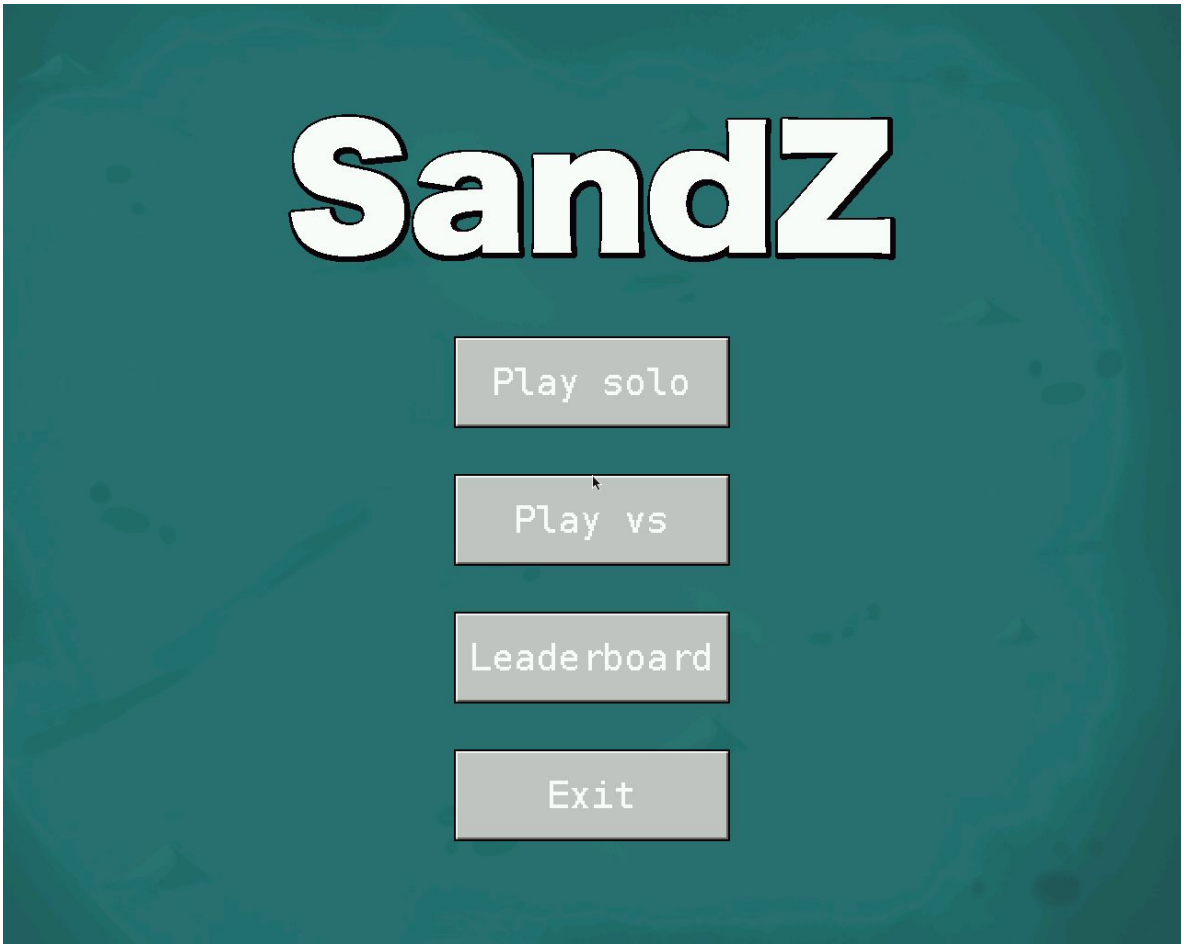
SandZ

O Top-Down Wave Survival em Minix

João Renato Costa Pinto, up201705547
Tiago Candeias Verdade, up201704003

Instruções

O programa abre com um menu, onde se encontra o nome do jogo e as seguintes opções selecionáveis através do rato.



-Opção “*Leaderboard* ”

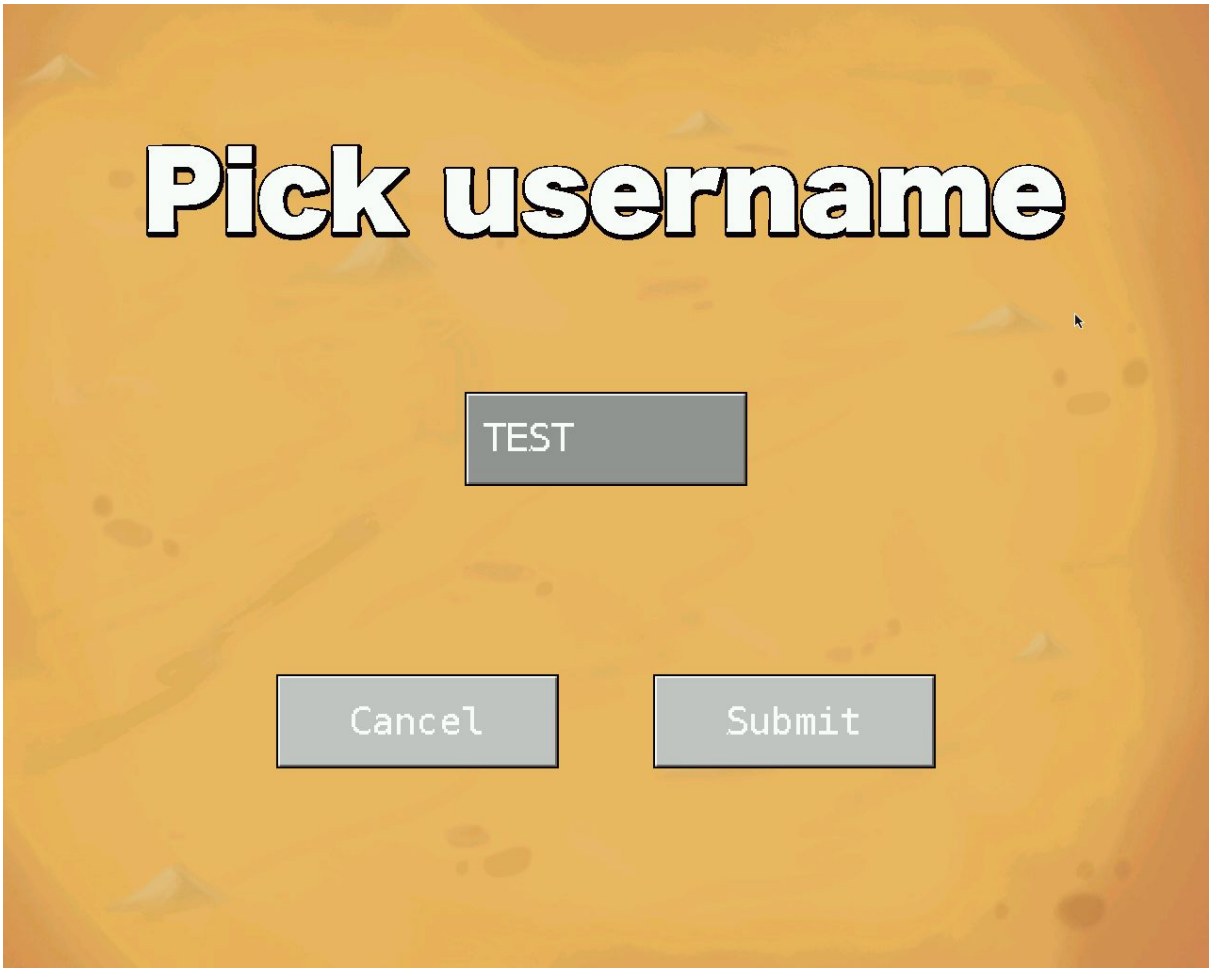
Quando selecionada a opção *Leaderboard,* o utilizador é redirecionado para um ecrã onde são mostradas as melhores pontuações obtidas pelos jogadores desde sempre, o seu nome e a data de quando foi obtida a pontuação. O utilizador pode regressar ao menu principal carregando no botão *return*.

-Opção “*Exit* ”

Quando selecionada termina a execução do programa.

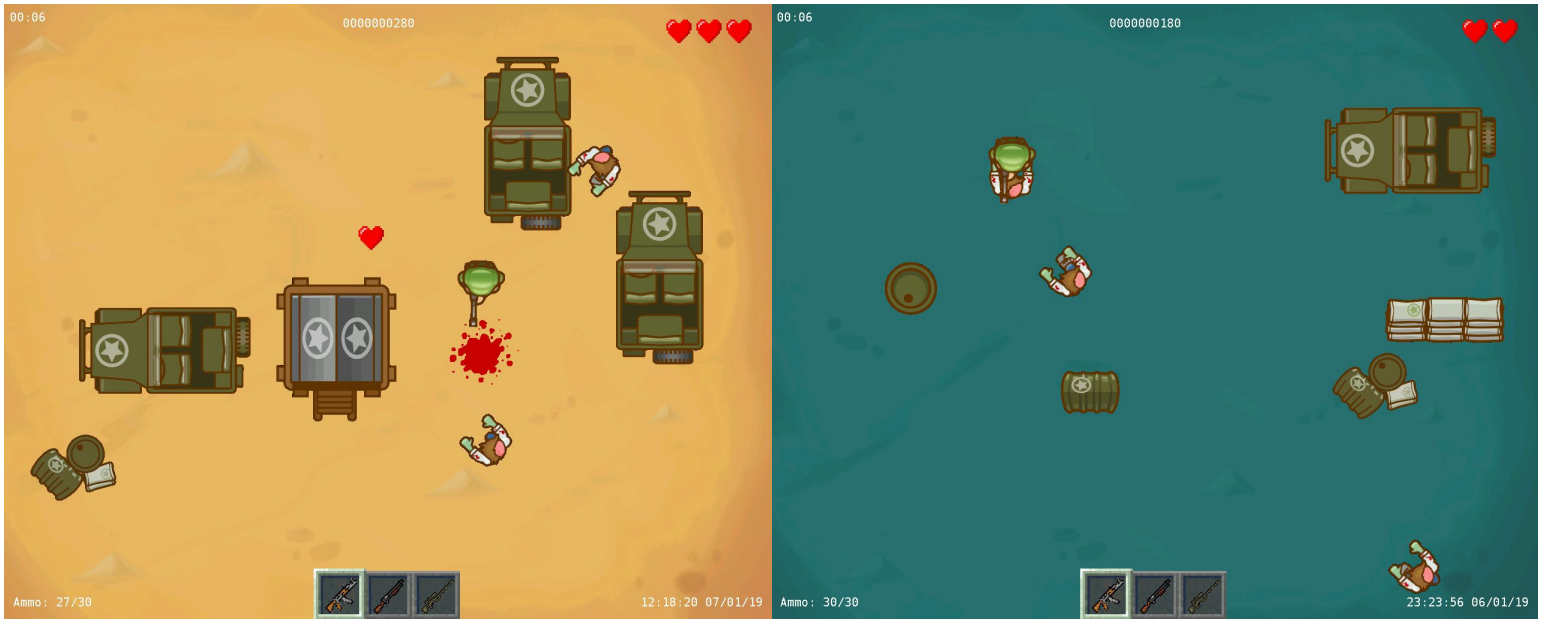
-Opção “*Play Solo*”

Reencaminha o utilizador ao ecrã de escolha de identificação, onde utiliza o teclado para escrever, apenas usando letras, o nome pretendido com um máximo de 25 letras. Pode retornar ao *main menu* através do botão “*Cancel”* ou dar início à partida com “*Submit”.*



Premindo “*Submit*” o utilizador é levado ao ecrã de jogo onde temos um soldado, o jogador controlado através do teclado, obstáculos, *zombies*, e a *UI* que consiste nas vidas atuais do jogador, arma selecionada, pontuação, tempo de jogo, munição disponível e data e hora atuais.

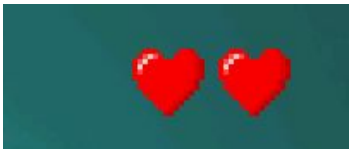
-Opção “*Game*”



O objetivo do jogo é simples, sobreviver o máximo de tempo possível matando o maior número de *zombies* ao passar de rondas com dificuldade crescente, de forma a obter a maior pontuação. O jogador pode colocar o jogo em pausa usando a tecla P do teclado, dando *display* do ecrã de pausa que tem o jogo pausado como fundo.

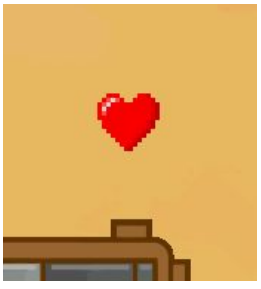
No ecrã do jogo há muitas informações de interesse para o jogador dentro das quais:

- Número de vidas: (Canto superior direito)

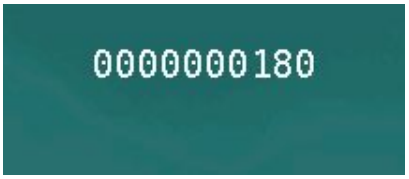


Nos quais é possível observar o número de vidas do jogador, as quais têm um máximo de 3 vidas e um mínimo de 0, sendo que ao atingir 0 vidas o jogo é terminado e o jogador é levado ao game over screen.

É possível o jogador restaurar alguma das suas vidas perdidas apanhando as vidas que vão aparecendo no mapa em coordenadas aleatórias.



- Pontos de jogo: (Centro superior)



Neste campo é possível observar os pontos que o jogador acumulou durante o jogo que está a decorrer.

Existem duas formas de ganhar pontos, sendo elas tempo de sobrevivência e eliminação de zombies.

Os pontos ganhos durante a sobrevivência têm em conta a estabilidade de jogo do player, por outras palavras o número de vidas que o player tem no presente. Recebendo mais pontos caso tenha mais vidas correntemente.

Quanto à eliminação de zombies o jogador ganha mais pontos por eliminação consoante a ronda em que está presente. Sendo que os zombies em rondas mais avançadas têm mais vida e são mais difíceis de eliminar, o jogador recebe mais pontos por eliminação em rondas maiores.

- Seleção de armas: (Centro inferior)

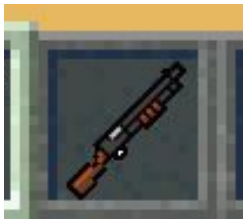
Nesta secção é possível visualizar as armas disponíveis para seleção e a arma equipada de momento. Cada arma tem as suas características específicas e cabe ao utilizador escolher a melhor arma para o momento. Nesta imagem é possível visualizar que a primeira arma está selecionada. O utilizador pode alterar a arma equipada através do keyboard usando os números 1, 2, 3, selecionando a arma respetiva.

Armas disponíveis:

1. **Assault Rifle**

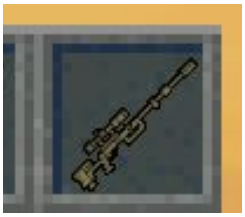
Arma selecionada por default no início, bastante simples. Tem carregadores de 30 balas, as quais viajam sempre na direção em que o player as disparou. Tem dano normal e não conseguem perfurar os zombies.

2. **Shotgun**



Arma selecionada através da tecla 2, excelente para combate mais próximo . Tem carregadores de 5 cartuchos, os quais consistem de 3 balas que viajam em forma de cone na direção em que o player as disparou. Tem dano normal e não conseguem perfurar os zombies.

3. Sniper



Arma selecionada através da tecla 3, excelente para combate ao longe . Tem carregadores de 3 balas, as quais viajam sempre na direção em que o player as disparou. Tem dano alto e consegue perfurar os zombies.

- Indicador de carregadores: (Canto inferior esquerdo)

Nesta parte do ecrã é possível visualizar à esquerda as munições restantes do carregador da arma selecionada e à direita a quantidade máxima do carregador.

Esta informação é útil para o jogador ter noção do número de balas restantes e planear quando deve recarregar a arma selecionada, premindo a tecla “R”.

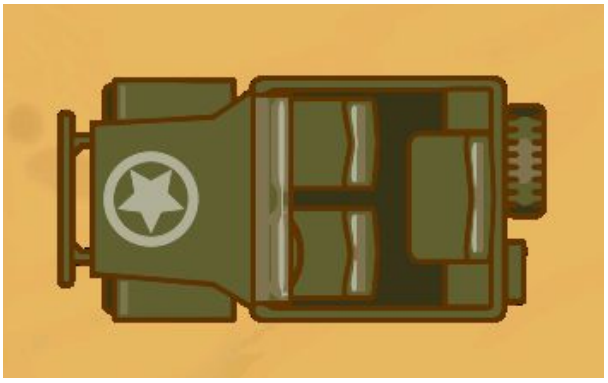
- Indicador de tempo jogado: (Canto superior esquerdo)

Temporizador muito básico que mostra o número de minutos, esquerda, e número de segundos, direita, a que o jogador está a jogar. Este temporizador não tem em conta o tempo passado no menu de pausa, sendo o tempo realmente ativo de jogo.

- Indicador de tempo presente: (Canto inferior direito)

Mostrador do tempo atual, mostrando as horas, minutos e segundos do dia presente, tal como o dia, mês e ano presente.

- Obstáculos:



No início do jogo são criados 5 obstáculos aleatórios, de uma seleção de 7 obstáculos diferentes, que são aleatoriamente distribuídos pelo mapa. O objectivo destes obstáculos são dificultar o movimento ao jogador, e também aos zombies.

- Zombies:

Os zombies são os únicos inimigos do jogador, movendo-se sempre em direção ao jogador, atacando o player quando estes entram em contacto com o player. O seu dano é sempre o mesmo, uma vida por ataque, no entanto a sua vida vai aumentando à medida que as rondas avançam. Zombies com a vida total não têm representado a sua barra de vida, mas ao levar qualquer quantidade de dano passam a mostrar a sua vida restante.

- Jogador:

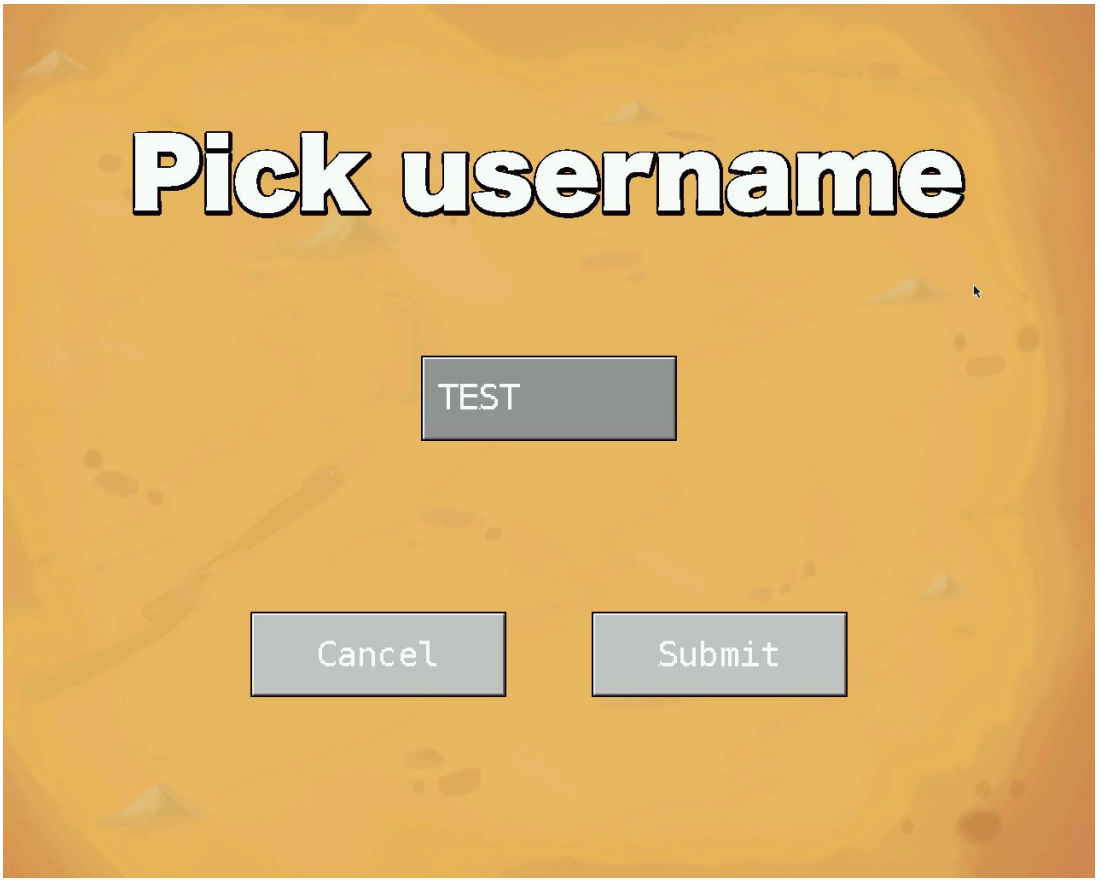
O jogador que é a personagem principal deste jogo, que consegue se movimentar nas 8 direções (N, NE, E, SE, S, SW, W, NW) tem como objectivo a sobrevivência, usando todas as ferramentas mencionadas anteriormente. O jogo acaba quando o player fica com zero vidas ou desiste. O jogador está sempre suscetível aos ataques dos zombies como é possível ver na imagem da esquerda, no entanto ao levar dano este tem um pequeno cooldown para levar dano novamente, este cooldown é representado através da coloração a vermelho do jogador, mostrando que este está invulnerável a dano.

-Opção “*Game pause*”

Aqui neste menu o utilizador pode fazer uma pequena pausa das suas horas de jogo e retomando ao mesmo através do botão “*Resume*”.
Pode também desistir do jogo através do botão “*Quit Game*”.

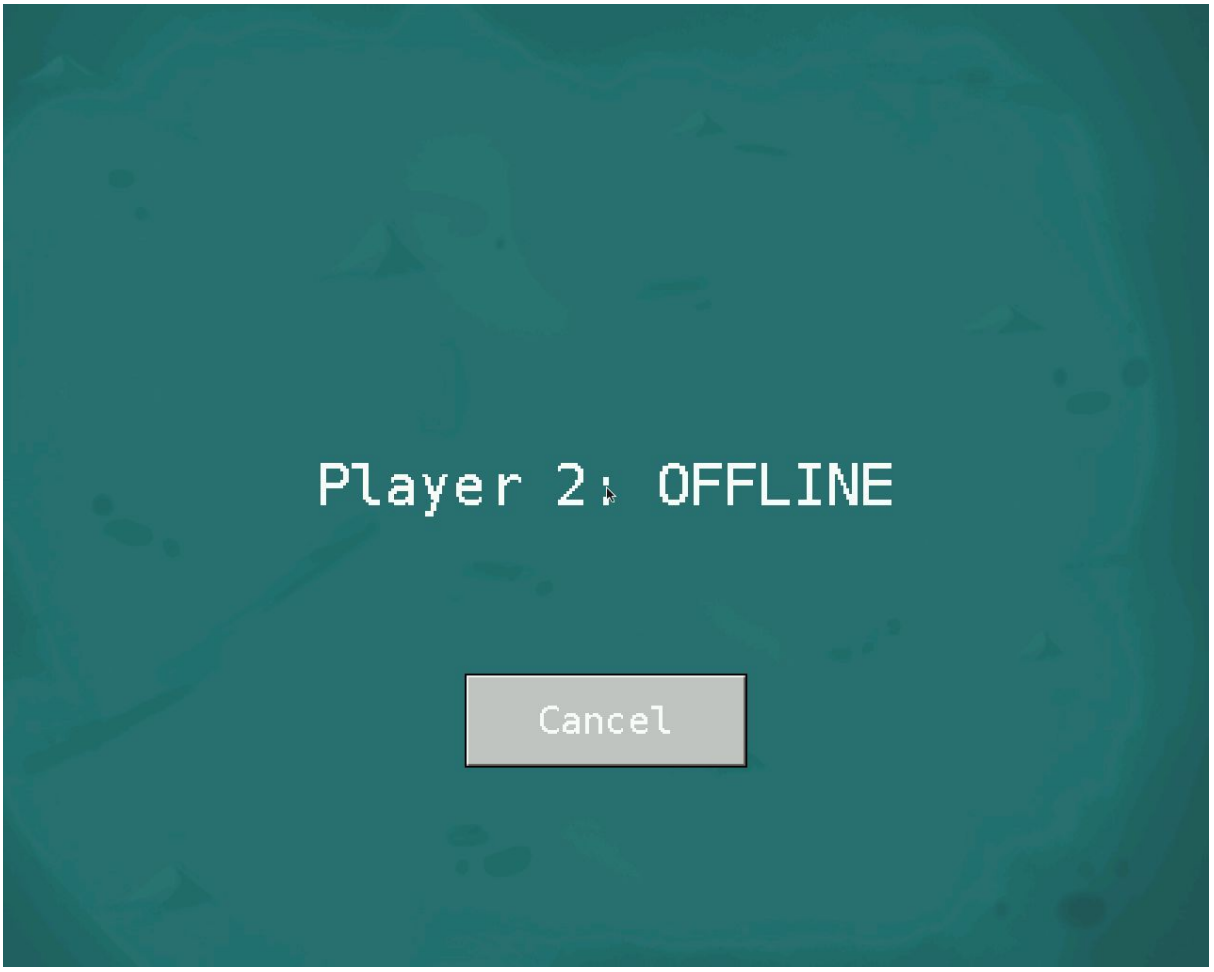
-Opção “*Play vs”*

Semelhantemente à opção “Play Solo” o utilizador é reencaminhado ao ecrã de escolha de identificação, onde utiliza o teclado para escrever, apenas usando letras, o nome pretendido com um máximo de 25 letras. Pode retornar ao *main menu* através do botão “*Cancel”* ou esperar pelo seu oponente através do botão “Submit”.



-Opção “*Wait player two”*

Neste menu ao jogador é dada a informação do estado do jogador 2, a qual pode ser “Offline”, o jogador não tem o jogo aberto, ou “Online” o jogador está com o programa aberto. O jogo é automaticamente começado quando ambos os jogadores estiverem neste ecrã. Caso o jogador não esteja mais interessado a esperar pelo outro jogador pode voltar ao menu screen usando o botão “Cancel”.



-Opção “*Game two”*

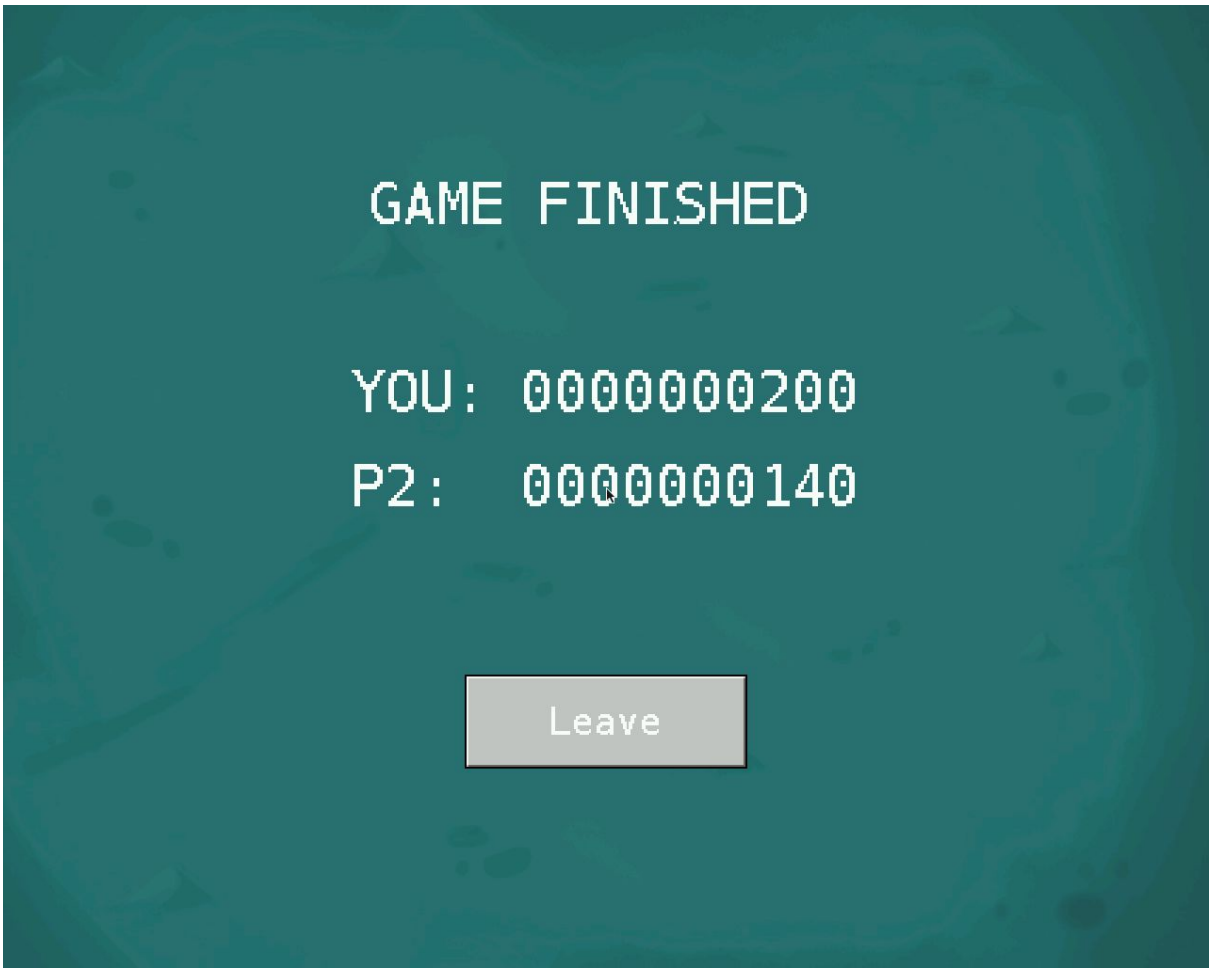
Jogo praticamente igual ao iniciado pela opção “Play Solo”, no entanto neste jogo o jogador está a competir contra um adversário. Cada um está a jogar o seu jogo, no entanto estão a tentar atingir uma maior pontuação que o adversário. É possível ver os scores tanto do jogador como do adversário em tempo real.

Como este jogo é online não é dada a opção de pausar o jogo.

-Opção “*Wait winner two*”

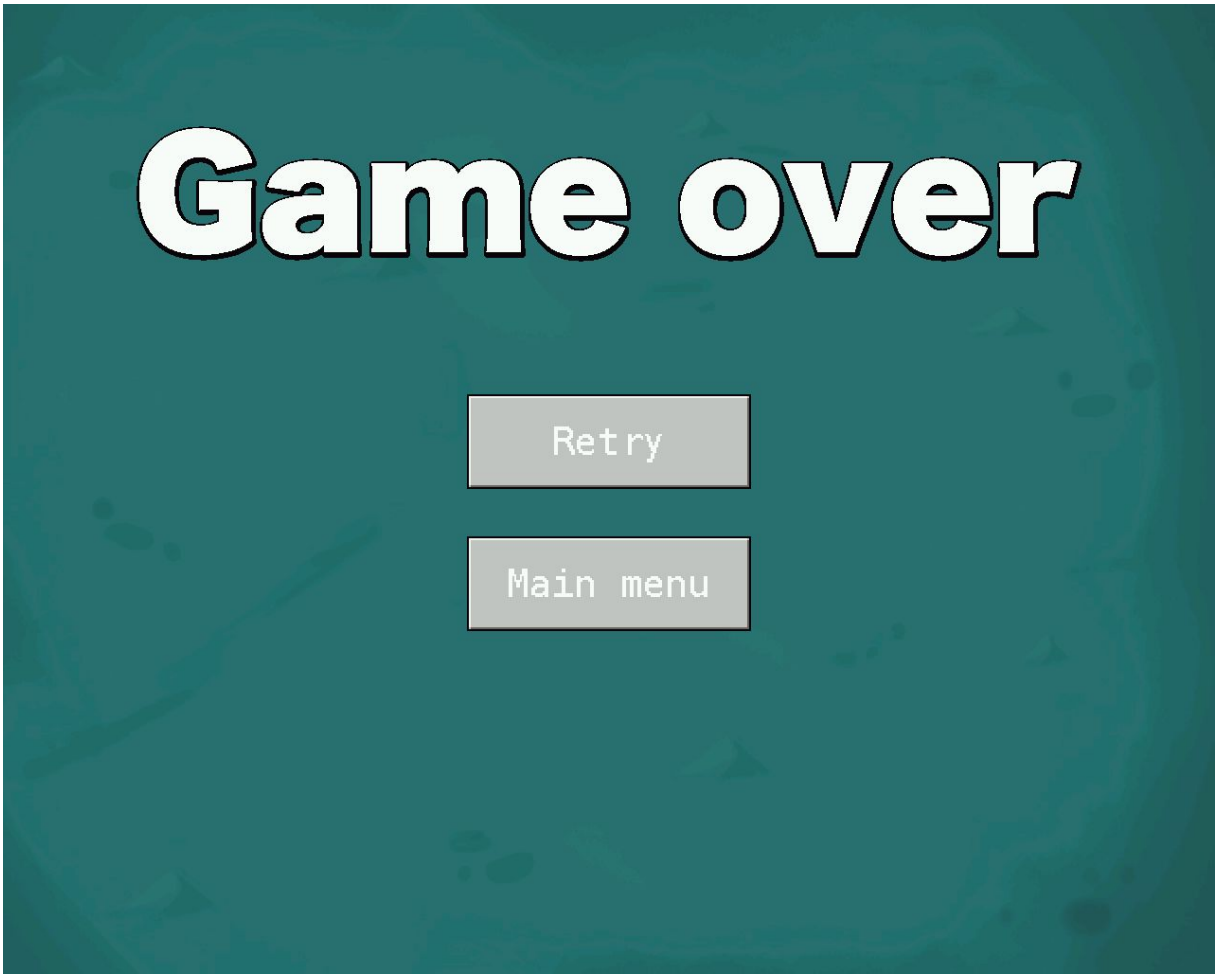
Neste menu o jogador pode visualizar o seu score obtido no jogo e o score do seu adversário em tempo real, tal como o estado do jogo do seu adversário.

Caso o jogador não tenha interesse nos resultados finais pode ir para o game over menu utilizando o botão “Leave”.


GAME FINISHED
YOU: 0000000200
P2: 0000000140
Leave

-Opção “*Game over*”

O *game over screen* é mostrado quando o jogador perde, isto é, quando fica com as suas vidas reduzidas a zero ou quando desiste através do ecrã de pausa.



A partir do *losing screen* pode regressar ao menu inicial com “*Main Menu”* ou imediatamente começar outro jogo premindo “*Retry*”.

Periféricos Utilizados

A seguinte tabela resume todos os periféricos utilizados pelo SandZ.

Periférico	Utilização	Interrupções
Timer	-Controlar a lógica do jogo e atualizar gráficos	Sim
KBD	-Controlar jogador e as suas ações e pausar jogo	Sim
Mouse	-Escolher opções nos menus e atualizar gráficos	Sim
Video Card	-Mostrar no ecrã os menus e o jogo	Não
RTC	-Controlar dia e noite -Mostrar hora e data atuais	Sim
Serial Port	-Comunicação entre computadores para *Multiplayer*	Sim

-Timer

O timer é utilizado para regular o *frame rate* do jogo.

Para isso usamos as suas interrupções com 60 interrupções por segundo. As funções usadas foram implementadas ao longo do Laboratório 2, e estão definidas em timer.c, timer.h e i8254.h. É utilizado também para controlar a duração de certas ações durante o decorrer do jogo (ex: Tempo de Invencibilidade do Jogador, Tempo de Recarregar Arma, etc…).

-KBD

O teclado é usado para controlar o jogador e as suas ações, ou seja, os movimentos (setas) e disparar as balas (barra de espaços), tal como para colocar o jogo em pausa (tecla P) e ainda permite receber como input o nome que o jogador quer usar (alfabeto).

As funções de subscrição e o *handlers* estão implementadas no kdc.h, kdc.c e i8042.h. São usadas no ficheiro proj.c

-Mouse

O rato é utilizado para navegar os menus, usando tanto o movimento como o botão esquerdo, e também para atualizar gráficos sempre que o rato se encontra por cima de um botão de menu, funcionando por interrupções gera uma sempre que a posição do rato é alterada ou é feito um clique.

As suas funções foram implementadas durante o Laboratório 4 e encontram-se no mouse.h, mouse.c e i8042.h.

-Placa Gráfica

A placa gráfica está encarregue dos gráficos do jogo, para isso usamos a resolução 1280x1024, com 16 bits por pixel (12^6 = 65536 cores), com RGB 5-6-5. O modo utilizado é o 0x11A.

No projeto é utilizado double buffering para manter os gráficos suaves e consistentes, os objetos do jogo não são da nossa autoria, mas sim objetos disponíveis com licença livre que achamos adequados.

A implementação está feita em video.h e video.c.

-RTC

O RTC é usado para obter a data e hora que mostramos no jogo, para guardar a data/hora da pontuação obtida e para escolher se é usado o *background* diurno ou noturno do jogo.

A implementação está feita nos ficheiros rtc.c e rtc.h.

-Serial Port

A porta de série é utilizada para comunicação entre computadores. Estes transmitem informação do estado do programa, se está online ou offline, e durante o jogo online transmitem a pontuação dos jogadores mutuamente.

Estas mensagens enviadas entre computadores seguem um padrão específico. Para transmitir o estado do programa, o estado, número identificado de 1 a 3, é envolvido por uma letra inicial de mensagem ‘s’ e por uma letra final ‘S’. O computador ao receber informação, se identificar alguma anomalia na mensagem descarta a mesma. O semelhante acontece para o envio de informação relativo aos pontos do jogador, marcando a letra inicial ‘p’ e a final ‘P’, estando entre estas os pontos do jogador.

A implementação está feita nos ficheiros serial_port.c e serial_port.h.

Estrutura e Organização do Código

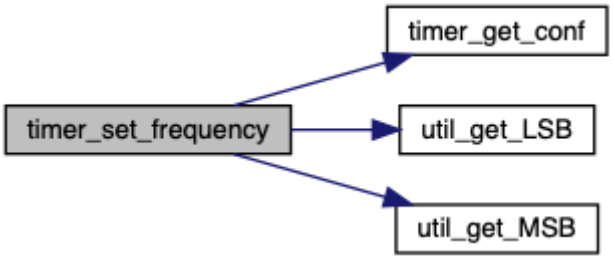
O código está estruturado em módulos, de forma a torná-lo mais legível e organizado. Os diagramas de chamada de funções foram gerados pelo Doxygen.

-Módulos dos periféricos

Os seguintes módulos contêm a implementação dos periféricos, tendo sido implementados durante os Laboratórios.

- Timer.c

Este módulo contém as funções que desenvolvemos na aula laboratorial 2, ou seja, as funções de subscrição do timer e os seus *interruption handlers.*

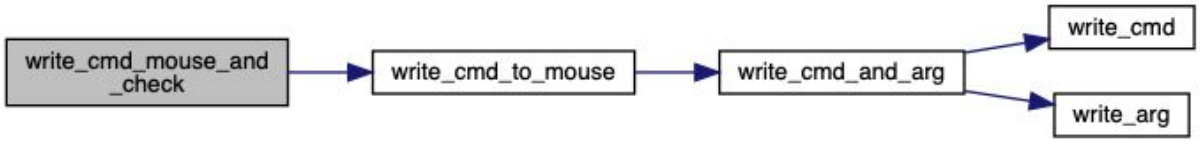


- Keyboard.c

Contém todas as funções de manipulação direta do teclado desenvolvidas no Laboratório 3.

- Mouse.c

Este módulo contém as funções de manipulação do rato implementadas no Laboratório 4.

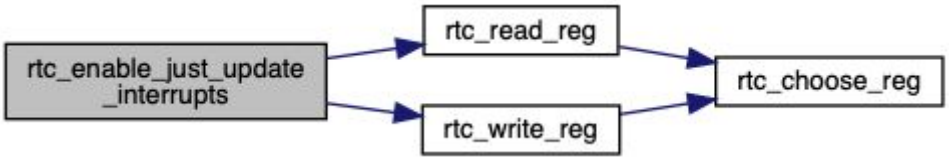


- Video.c

Neste módulo estão implementadas as funções da placa gráfica que desenvolvemos no Laboratório 5.

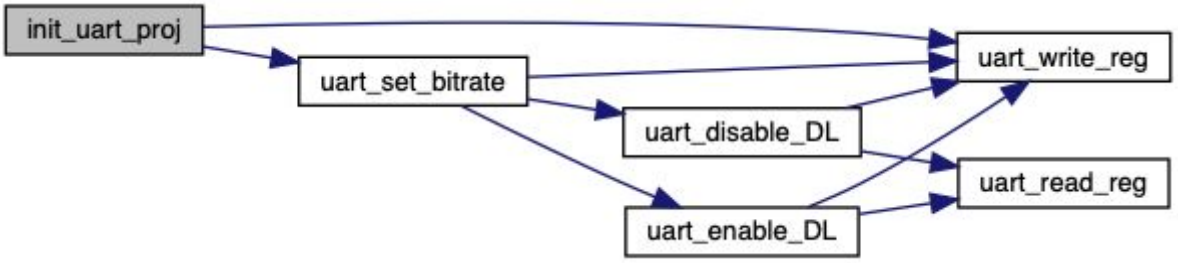
- RTC.c

Neste módulo estão implementadas as funções do Real Time Clock. Este módulo foi desenvolvido pelo Tiago Verdade.



- Serial_Port.c

Contém as funções de manipulação direta da porta de série. Este módulo foi desenvolvido pelo Tiago Verdade.



-Módulos relacionados com o jogo

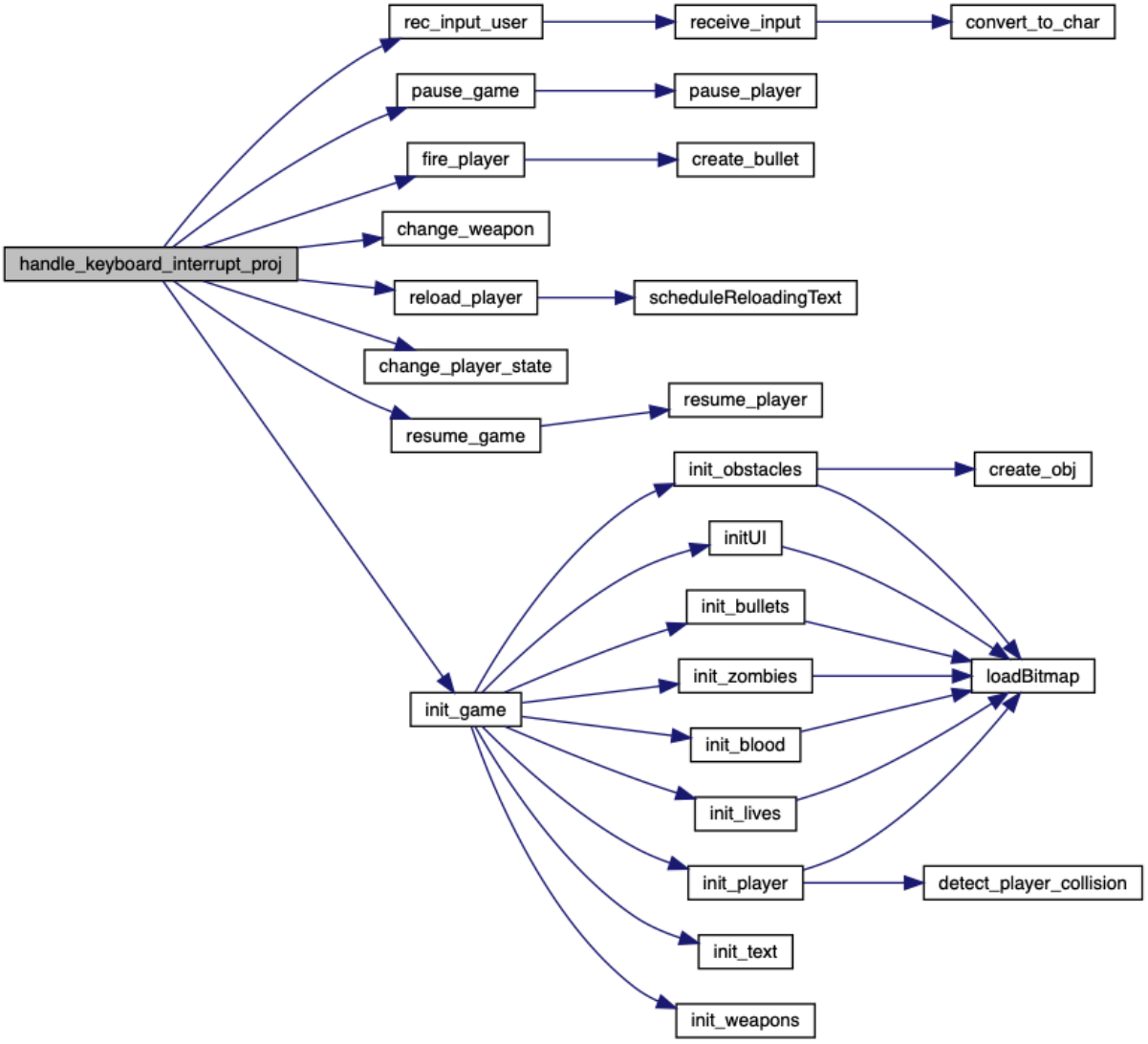
- Bitmap.c

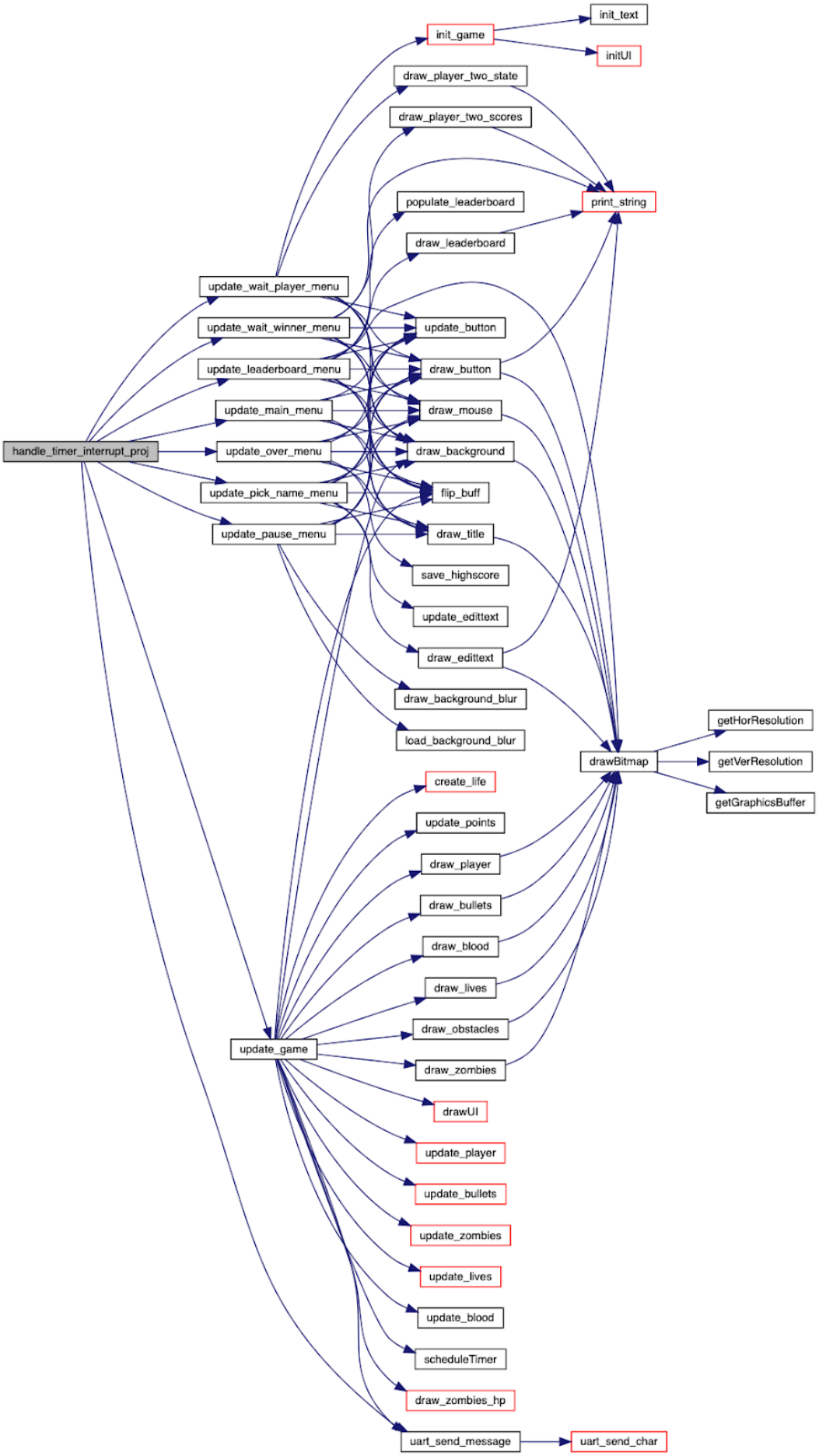
Este ficheiro contém as funções que nos permitiram carregar, desenhar e eventualmente apagar bitmaps. O código não é da nossa autoria, sendo este proveniente do blog do ex-aluno Henrique Ferrolho:

http://difusal.blogspot.pt/2014/09/minixtutorial-8-loading-bmp-images.html

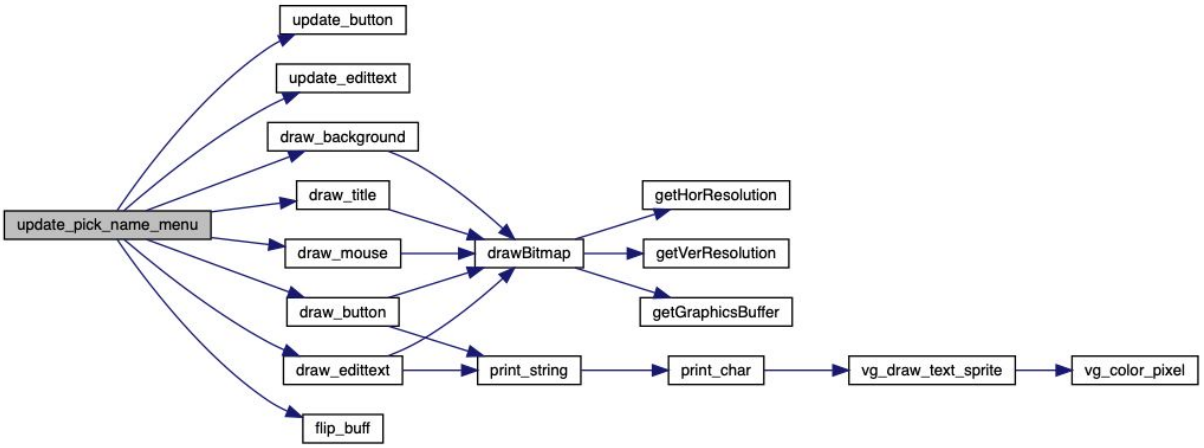
- Program.c

Este módulo contém as funções de controlo do programa (inicialização e interrupções) de forma a controlar o estado do programa (máquina de estados) e os respetivos eventos. Este módulo foi implementado por ambos.


handle_keyboard_interrupt_proj
rec_input_user
receive_input
convert_to_char
pause_game
pause_player
fire_player
create_bullet
change_weapon
reload_player
scheduleReloadingText
change_player_state
resume_game
resume_player
init_game
init_obstacles
create_obj
initUI
init_bullets
init_zombies
loadBitmap
init_blood
init_lives
init_player
detect_player_collision
init_text
init_weapons


handle_timer_interrupt_proj
init_game
init_text
initUI
draw_player_two_state
draw_player_two_scores
populate_leaderboard
draw_leaderboard
print_string
update_wait_player_menu
update_wait_winner_menu
update_leaderboard_menu
update_main_menu
update_over_menu
update_pick_name_menu
update_pause_menu
update_button
draw_button
draw_mouse
draw_background
flip_buff
draw_title
save_highscore
update_edittext
draw_edittext
draw_background_blur
load_background_blur
drawBitmap
getHorResolution
getVerResolution
getGraphicsBuffer
update_game
create_life
update_points
draw_player
draw_bullets
draw_blood
draw_lives
draw_obstacles
draw_zombies
drawUI
update_player
update_bullets
update_zombies
update_lives
update_blood
scheduleTimer
draw_zombies_hp
uart_send_message
uart_send_char

- Main_menu.c, Leaderboard_menu.c, Pick_name_menu.c, Over_menu.c, Wait_player_menu.c e Wait_winner_menu.c



Estes ficheiros possuem as implementações dos diferentes menus e respetivas funções de manipulação. Apesar de serem menus diferentes com diferentes funções a sua implementação é muito semelhante, constituindo principalmente de buttons, edit_texts e texts. Estes módulos foram implementados pelo Tiago Verdade.

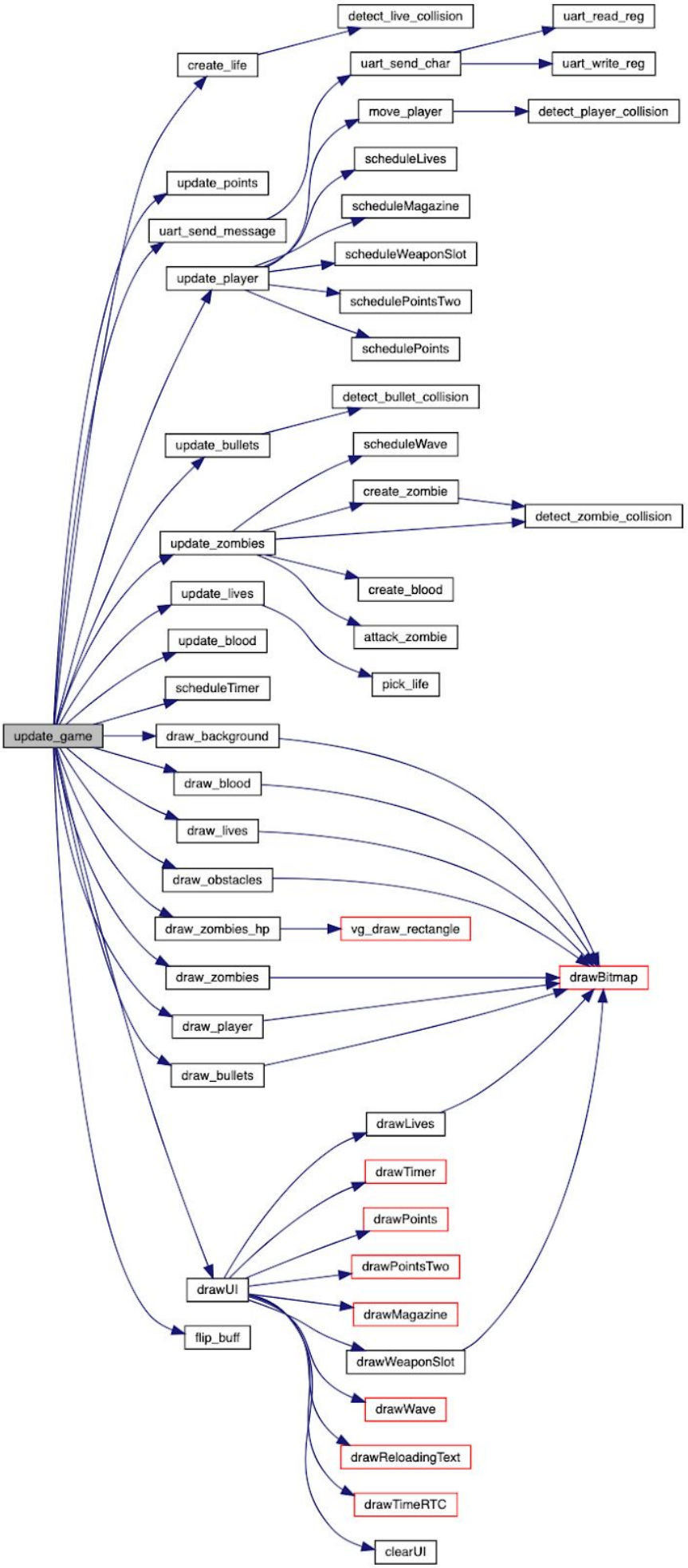
- Pause_menu.c

Este ficheiro possui uma implementação muito parecida aos outros menus, no entanto há um pormenor que acredito que deva ser realçado. Em todos os outros menus o background é igual, dependendo de ser de dia ou noite muda, mas só alternam entre esses dois backgrounds. No entanto o menu de pausa utiliza o ecrã anteriormente mostrado no jogo (game.c) e faz alterações a essa mesma “imagem”, com o objetivo final de fazer o efeito de imagem chamado *blur*, ao screen. Para fazer este efeito estamos a usar um algoritmo chamado box blur, o qual para cada pixel vai ver a média das cores dos píxeis à sua volta num “raio” r. Assim para cada pixel faz-se a média das cores contidas no quadrado (x-r, y-r) (x-r, y+r) (x+r, y-r) (x+r, y+r) e atribui-se esse valor ao mesmo. A implementação “básica” do algoritmo tem uma complexidade O(n*r^2), sendo n o número de píxeis. No nosso projeto, usando um raio pequeno (r = 10) e com o background que tem (1280*1024) 1,310,720 píxeis, seriam necessárias 131,072,000 operações para gerar o background do pause menu, número demasiado alto quando a transição entre o jogo e o menu é feita em 16.7ms. Assim tivemos que otimizar o algoritmo de box blur usando um algoritmo chamado “sum 2d array” o qual consegue calcular a soma de um sub2D array em tempo constante, tornando o algoritmo do box blur em O(n), o suficiente para tornar a transição entre ecrãs “smooth”.

Este módulo foi implementado pelo Tiago Verdade.

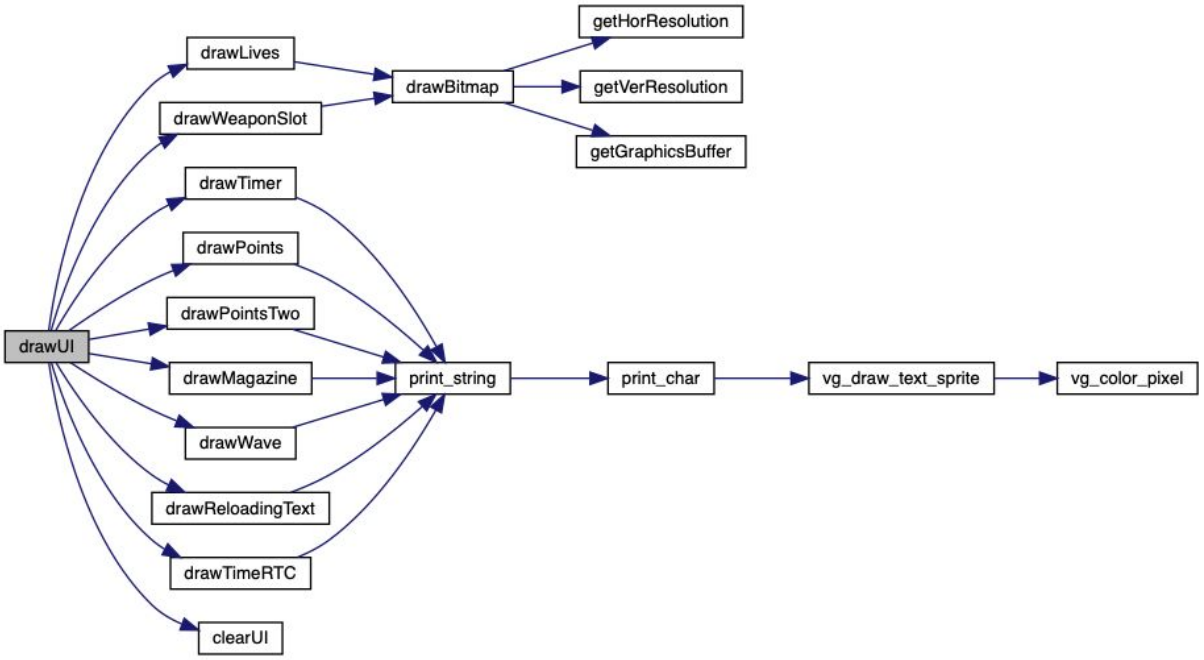
- Game.c

Este ficheiro contém as funções relativas ao jogo e a sua lógica em ambos os modos de jogo e manipulação do jogo (inicialização, atualização, pausa e retoma). Este módulo foi implementado por ambos.


update_game
create_life
detect_live_collision
uart_send_char
uart_read_reg
uart_write_reg
move_player
detect_player_collision
scheduleLives
update_points
scheduleMagazine
uart_send_message
scheduleWeaponSlot
update_player
schedulePointsTwo
schedulePoints
detect_bullet_collision
update_bullets
scheduleWave
create_zombie
detect_zombie_collision
update_zombies
create_blood
update_lives
attack_zombie
update_blood
pick_life
scheduleTimer
draw_background
draw_blood
draw_lives
draw_obstacles
draw_zombies_hp
vg_draw_rectangle
drawBitmap
draw_zombies
draw_player
draw_bullets
drawLives
drawTimer
drawPoints
drawPointsTwo
drawUI
drawMagazine
flip_buff
drawWeaponSlot
drawWave
drawReloadingText
drawTimeRTC
clearUI

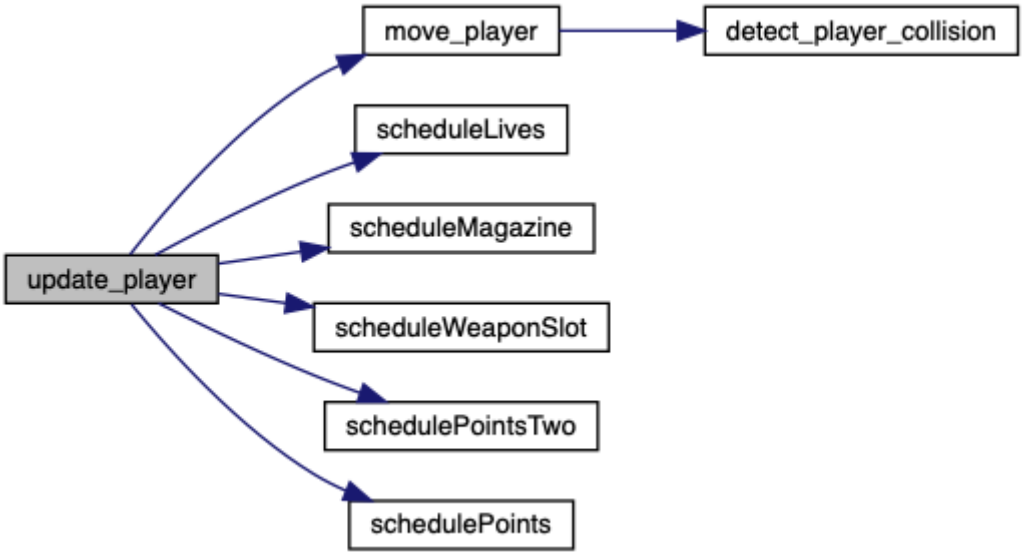
- Ui_game.c

Neste módulo estão implementadas as funções de controlo da *UI* do jogo, ou seja, as funções de *schedule* e *drawing* da *interface* (ronda, pontuação, vidas, texto, data, munição, etc...), implementadas por João Pinto (40%) e Tiago Verdade (60%).



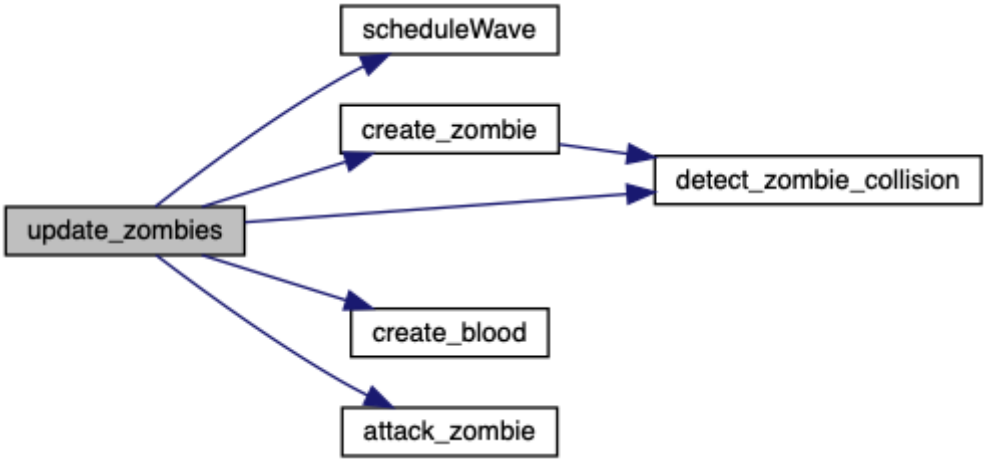
- Player.c

Neste módulo estão todas as funções de controlo do jogador e as suas ações, e interação com o terreno. Foi implementado por Tiago Verdade (80%) e João Pinto (20%).



- Zombie.c

Neste ficheiro estão todas as funções de manipulação dos objetos *zombie* e a sua interação com o jogador e terreno. Este módulo foi realizado por ambos.



- Blood.c

Neste módulo encontram-se as funções de manipulação dos objetos *blood* que representam o sangue deixado pela morte de um *zombie*. Este módulo foi implementado por João Pinto.

- Weapon.c

Este módulo tem as funções de manipulação das armas disponíveis ao jogador. Este módulo foi implementado por João Pinto.

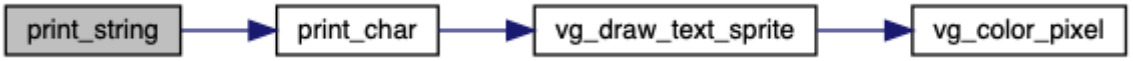
- Obstacle.c

Neste ficheiro estão todas as funções de manipulação dos objetos *Obstacle* gerados aleatoriamente pelo mapa de forma a tornar o jogo menos repetitivo, foi implementado pelo Tiago Verdade.

- Life.c

Neste ficheiro estão as funções de controlo dos objetos *life* que representam as vidas que aparecem periodicamente e disponíveis para o player. Este módulo foi realizado pelo João Pinto.

- Text.c

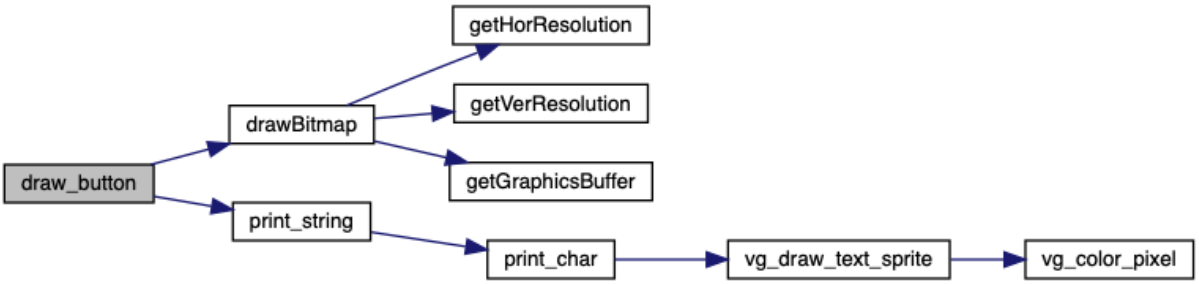


Neste módulo encontram-se as funções que permitem o desenho de texto quer na *Interface* quer nos menus. Para atingir este objetivo utilizamos um xpm, criado no GIMP, que contém todas as letras entre os valores ASCII 32 ‘ ‘ e 126 ‘~’, num tamanho de letra 20 e usando uma fonte monospace sem anti-aliasing. Como as letras são monospace a largura de cada letra é a mesma, então para escrever uma letra em específico simplesmente era preciso ver o seu valor (ASCII - 32)*char_width para saber o x inicial da letra no xpm e daí fazer draw do carácter.

Utilizando um tamanho de fonte pequeno é possível imprimir no ecrã as letras com diferentes tamanhos. O algoritmo usado para isto foi em vez de só imprimir um pixel por pixel do xpm, imprimir um quadrado de píxeis 2x2, 3x3, por cada pixel no xpm. Tornando possível escrever os textos com diferentes tamanhos usando um único xpm. É possível ver estes tamanhos usados na UI_game, nos botões e nos wait screen do multiplayer.

Este módulo foi implementado pelo Tiago Verdade.

- Button.c



Neste módulo encontram-se as funções que permitem o desenho de botões nos menus. Como este é um elemento fulcral para os menus, foi decidido tornar a criação dos botões o mais simples possível. Passando a uma função construtora as coordenadas do botão, o texto do botão, o tamanho do texto e a função a chamar quando este é premido temos um botão gerado. Para o botão funcionar corretamente é simplesmente preciso chamar a função “update_button” a cada frame.

Este módulo foi implementado pelo Tiago Verdade.

- Edit_text.c

Neste módulo encontram-se as funções que permitem ter uma caixa de texto que recebe input de texto do utilizador. Esta caixa de texto suporta as letras do alfabeto e o clear num máximo de 25 letras. Devido à caixa de texto não ser larga o suficiente para conter tantas letras teve de se criar uma “Listview” horizontal que mostrasse apenas as últimas letras escritas, sem perder a informação das letras iniciais que não estão a ser desenhadas. Este módulo foi implementado pelo Tiago Verdade.

- Mouse_proj.c

Este módulo contém as funções responsáveis pela manipulação do rato dentro do projeto, foi implementado pelo Tiago Verdade.

Conclusão

A disciplina de LCOM foi uma das que sentimos que mais nos desafiou ao longo do ano, quer com os labs quer com o projeto. Para os labs no início sentimos que andamos um bocado “perdidos”, que muitas vezes não estávamos a entender a lógica por de trás do que nos era pedido para fazer. No entanto com o passar do tempo e depois de muitas pesquisas sobre os vários assuntos da disciplina, conseguimos “acompanhar” a matéria que estava a ser dada.

Apesar da dificuldade que sentimos também achamos que foi uma das disciplinas mais divertidas, principalmente no projeto, e das quais nos vamos lembrar por mais tempo.

Apêndice

Instalação

Para instalar o jogo é necessário mudar de directório para o root do projeto.

```
cd labs/lcom1819-t4g13/proj
```

Dar as devidas permissões ao programa

```
lcom_conf add conf/proj
```

Compilar o programa

```
make
```

Correr o programa

```
lcom_run proj
```