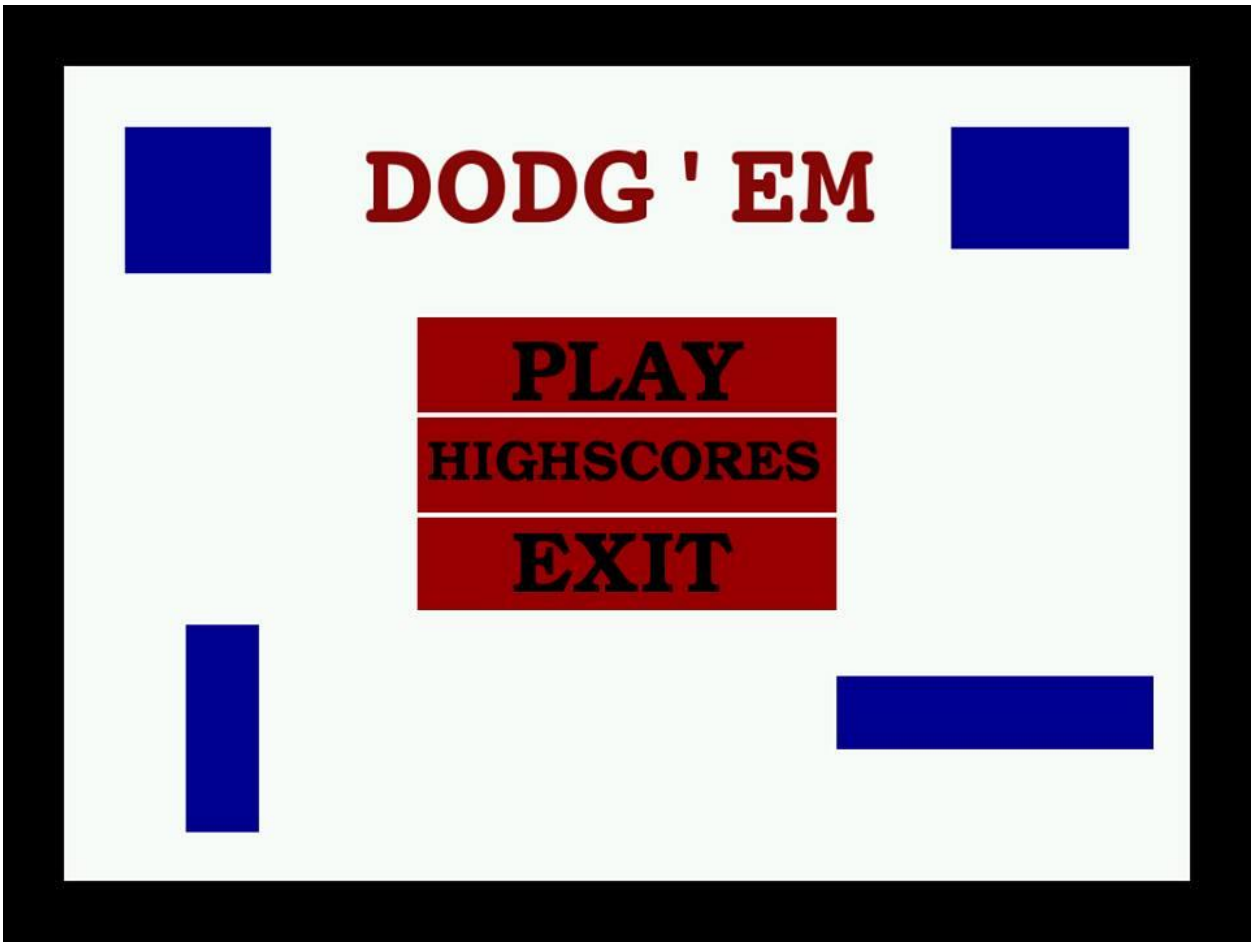
Faculdade de Engenharia da Universidade do Porto

Laboratório de Computadores
2014/2015 1º Semestre

DODG'EM



Turma 5 Grupo 13

Autores:

Daniel Silva Reis up201308586@fe.up.pt
João David Gonçalves Baião up201305195@fe.up.pt

Índice de Ilustrações:

Índice

Resumo

O nosso projeto, desenvolvido para a cadeira de laboratório de computadores, chama-se DODGEM e é uma versão melhorada de um jogo que facilmente se encontra pela internet cujo nome é “Desafio dos 19 segundos” ou “You’re a genius”.
Decidimos fazer uma versão melhorada do jogo em que incluimos “power ups” e um menu com os highscores.

O objetivo deste jogo é aguentar o máximo de segundos possíveis sem colidir com os objetos e sem sair da área de jogo com ou sem uso de poderes. Estes poderes podem parar os objetos em movimento, tornar o quadrado de jogo invencível e tornar os objetos mais lentos.
Para realizar este projeto, tivemos de implementar os periféricos quase de raíz, o que nos permitiu aprofundar os nossos conhecimentos a nível de hardware de periféricos.

Software utilizado:

- Sistema Operativo Minix através de uma máquina virtual criada como software VMware Player;
- Eclipse para programar todo o projecto;
- Redmine (SVN) da FEUP para a gestão das versões do nosso projecto;
- Doxywizard para gerar a documentação Doxygen;
- GIMP Image Editor para criar e editar as imagens;

Breve descricão do Programa:

O nosso jogo, DODGEM é um jogo simples onde temos de evitar colidir com os objetos que se encontram em movimento pela área de jogo usando poderes para ser mais fácil esquivar dos objetos (usando teclas que ativam os poderes) e o rato para movimentar o quadrado de jogo.

Antes de iniciar o jogo é apresentado um menu onde podemos escolher com o rato uma das opções:

1. Play
2. HighScores
3. Exit

Quando selecionamos a opção 1, entramos no menu de jogo onde podemos começar a jogar.

Se escolhermos a opção 2, entramos no menu onde podemos escolher ver as pontuações mais altas de cada um dos modos de jogo. Com border (em que o jogo se torna ligeiramente mais fácil) e sem border.

Na terceira opção, saímos do jogo.

Arquitectura do programa:

Módulos:

- main: Máquina de estados, possui a máquina de estados principal do jogo, que controla o menu principal do jogo.
- DODGEM: Possui todos os menus e funções de suporte aos mesmos.
- bitmap: As funções que alteram e desenham os bitmaps nos buffers.
- graphics: Funções para inicialização do modo de video e dos três buffers usados (video_mem, Video Buffer e Triple Buffer).
- i8042 e i8253: Possui macros a serem usadas pelos periféricos.
- keyboard: subscrição do teclado e funções para uso do mesmo.
- mouse: subscrição do rato e funções para manipulação do mesmo.
- RTC: subscrição do RTC e função que retorna valores de um certo registo do mesmo.
- timer: funções para uso do timer.
- utilities: macros para uso do programa.
- video_gr: Inicialização e saída de modo de video.

Estruturas de dados utilizadas:

- DODGEM: struct que guarda os bitmaps usados, as areas (rectangle) do programa, os FPS usados e os valores dos irqs usados.
- rectangle: struct que possui as coordenadas de todos os retangulos dos menus e também dos objetos de jogo, sendo que estes também possuem uma direção do movimento.
- POWER: struct que possui informações sobre os poderes utilizados pelo jogo, incluindo a quantidade de energia.
- SCORE: struct que guarda os valores do score atual e do melhor score.
- PLAYER: struct que possui o nickname do utilizador, o score por ele feito e a data de submissão.
- Bitmap: estrutura para criar, alterar e destruir bitmaps.
- Vetores de PLAYER que guardam a informação das 4 melhores pontuações guardadas (variáveis globais).

Periféricos utilizados:

- Placa gráfica: em modo de vídeo, para a interface gráfica do nosso jogo;
- Teclado: através de interrupções, para navegar nos menus e escrever o nome de utilizador aquando de guardar as pontuações;
- Timer: para controlar o processamento e fluxo de eventos e para contar o tempo de jogo;
- RTC: para visualizar a data nos highscores;
- Rato: para navegar nos menus e controlar o quadrado de jogo;

À exceção do RTC todos os periféricos são usados por interrupções.

Fases de implementação do projeto:

Demo:

Para começar o projeto, copiamos todo o trabalho realizado nos labs até à data realizados, uma vez que estes possuíam grande parte das funções para implementação dos periféricos no programa. Tendo já os periféricos, criamos funções para manipulação do modo

de vídeo, juntamente com o rato, teclado e timer e começamos então pelo menu principal. As interrupções do timer (a 60 fps) desenhavam o menu principal e o bitmap correspondente ao rato por cima, as interrupções do rato alteravam as coordenadas do mesmo (que estavam guardadas na struct MOUSE em mouse.h) e o keyboard era responsável pela saída do programa, sendo que sempre que a ESC key era premida o programa terminava, sendo este o resultado apresentado por nós na Demo de dia 15/12/2014.

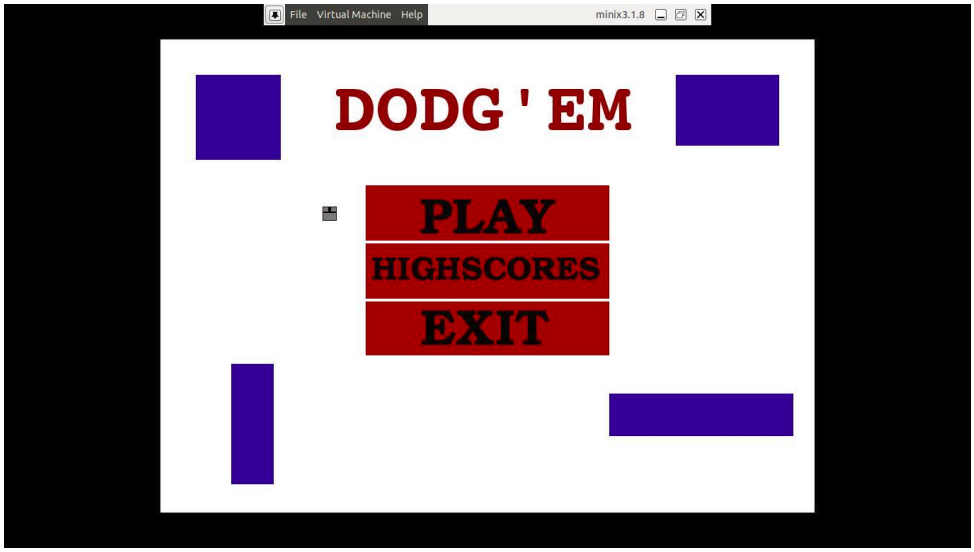


Ilustração 1: Main Menu

Maquina de estados principal:

Para o controlo do estado do jogo usamos uma máquina de estados correspondente ao menu principal. Essa máquina de estados encontra-se no main.c e serve para controlar a localização do programa em certo momento. Possui 3 estados diferentes correspondentes ao valor da variável MainOption, sendo 1 o menu de Jogo, 2 o menu de HighScores, 3 corresponde à saida do programa e quando se encontra a 0 corresponde ao menu principal.

Movimento e colisões:

Após a demo criamos grande parte das imagens utilizadas pelo programa através do GIMP e começamos a implementar o sistema de jogo. Colocamos os objetos azuis a moverem-se e a reconhecerem colisão com as paredes de jogo. O sistema de movimento dos objetos funciona da seguinte maneira:

Os objetos, pertencentes à struct “rectangle”, deslocam-se sempre na diagonal com um declive de 45º e portanto o seu descolamento em x e y é igual ao longo do tempo. Usamos uma variável “vel”, igual para todos os objetos, que corresponde ao movimento em x e y por

cada frame (dos 60 fps) e uma variável “direction” que varia entre 1 e 4, característica de cada objeto, que indica o quadrante da direção que o objeto possui. Quando o objeto colide com a área de jogo, este muda de direção.

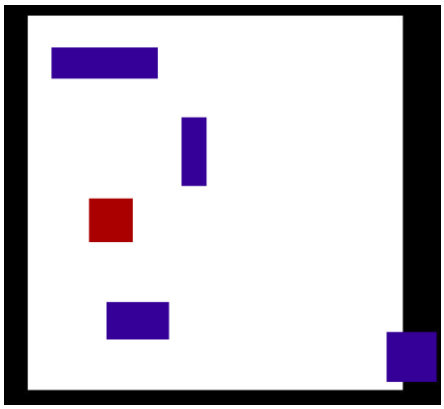


Ilustração 2: Área de jogo

A área de jogo dos objetos azuis não corresponde à área branca, sendo um pouco maior, o que possibilita que estes abranjam uma superfície maior da área branca.

Quando conseguimos colocar os azuis a mover-se corretamente fizemos com que o quadrado principal fosse controlado a partir das coordenadas do rato e criamos uma função muito simples e perfeitamente funcional para verificar colisão entre os objetos azuis e o de jogo, sendo esta função (“int CheckColisionObj(rectangle * Objeto)”) chamada em todos os frames, para todos os quadrados. Caso fosse verificada colisão o jogo terminava e o utilizador poderia escolher jogar novamente. Estava então o jogo em si funcional.

Pontuações:

A partir deste momento começamos a trabalhar nas pontuações. As pontuações correspondem ao tempo que o jogador aguenta sem colidir com os objetos ou a fronteira da área de jogo. Para tal convertemos os valores dados pelo timer para segundos e centesimas. A cada frame mostramos esses valores no ecrã a partir de uma função para desenhar algarismos de um bitmap de algarismos criado por nós no GIMP. A partir do momento em que o jogador colide, as variáveis da struct “SCORE” que possuem os valores das pontuações deixam de ser atualizadas e caso sejam maiores que o melhor score feito desde o início do programa substituem a melhor pontuação guardada

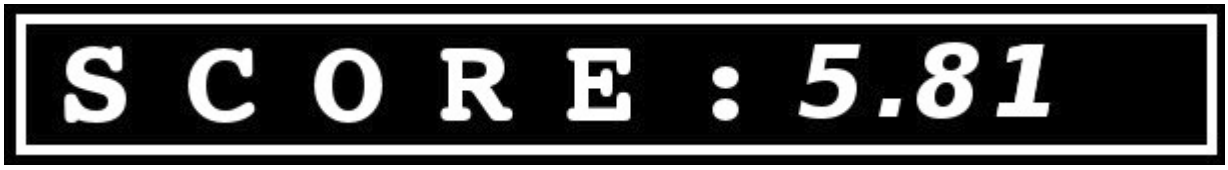


Ilustração 3: Pontuação atual

Ilustração 4: Melhor pontuação

Poderes:

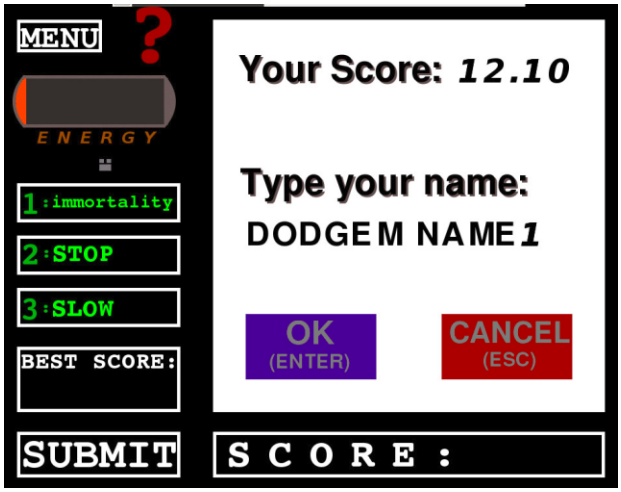
Para melhorar o jogo e para ajudar o jogador a aguentar mais tempo durante o jogo criamos poderes, poderes estes que não existiam no jogo original “You’re a genius”. O utilizador, através das teclas ‘1’, ‘2’ e ‘3’. Com a tecla 1 o utilizador fica imortal por um segundo e o quadrado altera para verde. Para que tal funcione, a struct “POWER” guarda uma variável “invencibilidade” que quando diferente de 0, o programa não verifica a colisão com os objetos. A tecla 2 faz com que os objetos parem de se mover, ou seja, quando a variável “stopMovement” é diferente de 0 as coordenadas dos objetos não são alteradas durante um segundo. A tecla 3 faz com que a variável “vel” seja reduzida para metade durante três segundos e, consequentemente, os objetos movem-se com metade da velocidade.

Para evitar que o utilizador “abuse” dos poderes, implementamos um sistema de energia, que corresponde a uma variável “Energy” que possui a quantidade de energia que o utilizador possui e varia de 0 a 100, sendo que cada poder tem um certo custo e a energia regenera 2 pontos por segundo.

Ilustração 5: Energia e Poderes

Guardar pontuações em ficheiros:

Para terminar o essencial do projeto tivemos de fazer uma maneira para guardar as pontuações registadas. Para tal criamos um menu para submeter a melhor pontuação registada.



Este menu mostra a melhor pontuação obtida registada para guardar e pede o nome do utilizador, nome este que deve ser escrito com o teclado e pode possuir algarismos, letras e espaços. À medida que o utilizador vai escrevendo, os caracteres vão aparecendo no ecrã.

Ilustração 6: Menu para submição de scores

Quando o utilizador guarda a pontuação, caso esta seja maior do que alguma registada até ao momento, um objeto da struct “PLAYER” que possui o nome do jogador, a pontuação e a data (obtida através do RTC) é adicionado ao array das pontuações na posição correta, sendo que este array se encontra ordenado.

Os arrays que possuem as 4 melhores pontuações (players_border e players_noborder) são então utilizados para mostrar ao utilizador no menu “highscores” as melhores pontuações jogadas com ou sem barreiras.

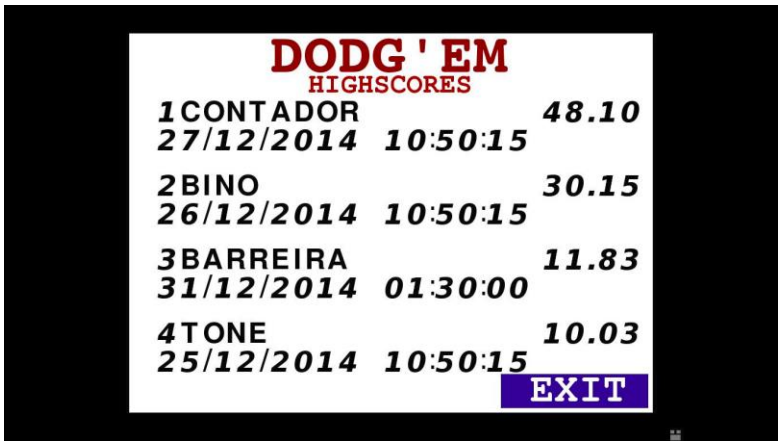


Ilustração 7: Lista de Pontuações

Quando o programa termina estas pontuações são guardadas no ficheiro “scores.txt” (void saveScores()) e são lidas quando o programa é iniciado (int loadScores()).

Extras:

Ao longo do desenvolvimento do programa foram implementadas outras funcionalidades não referidas anteriormente, tais como a possibilidade do jogador jogar com barreiras e portanto não perder ao tocar na fronteira de jogo e um menu de ajuda que mostra ao utilizador uma pequena descrição do jogo.

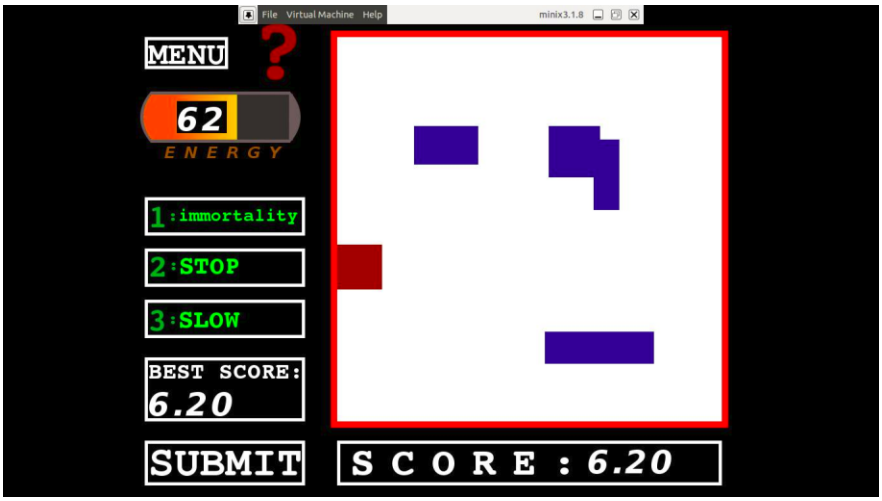


Ilustração 8: Jogo com barreira

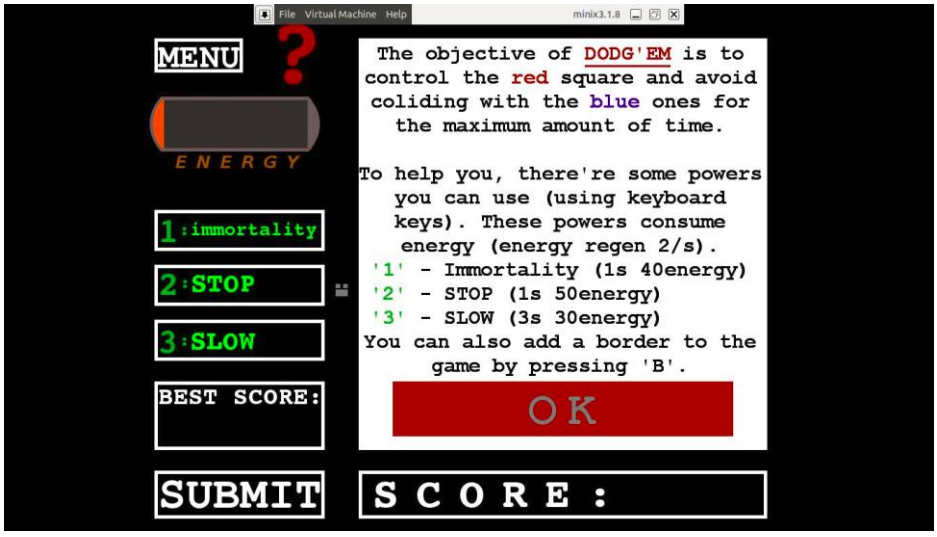
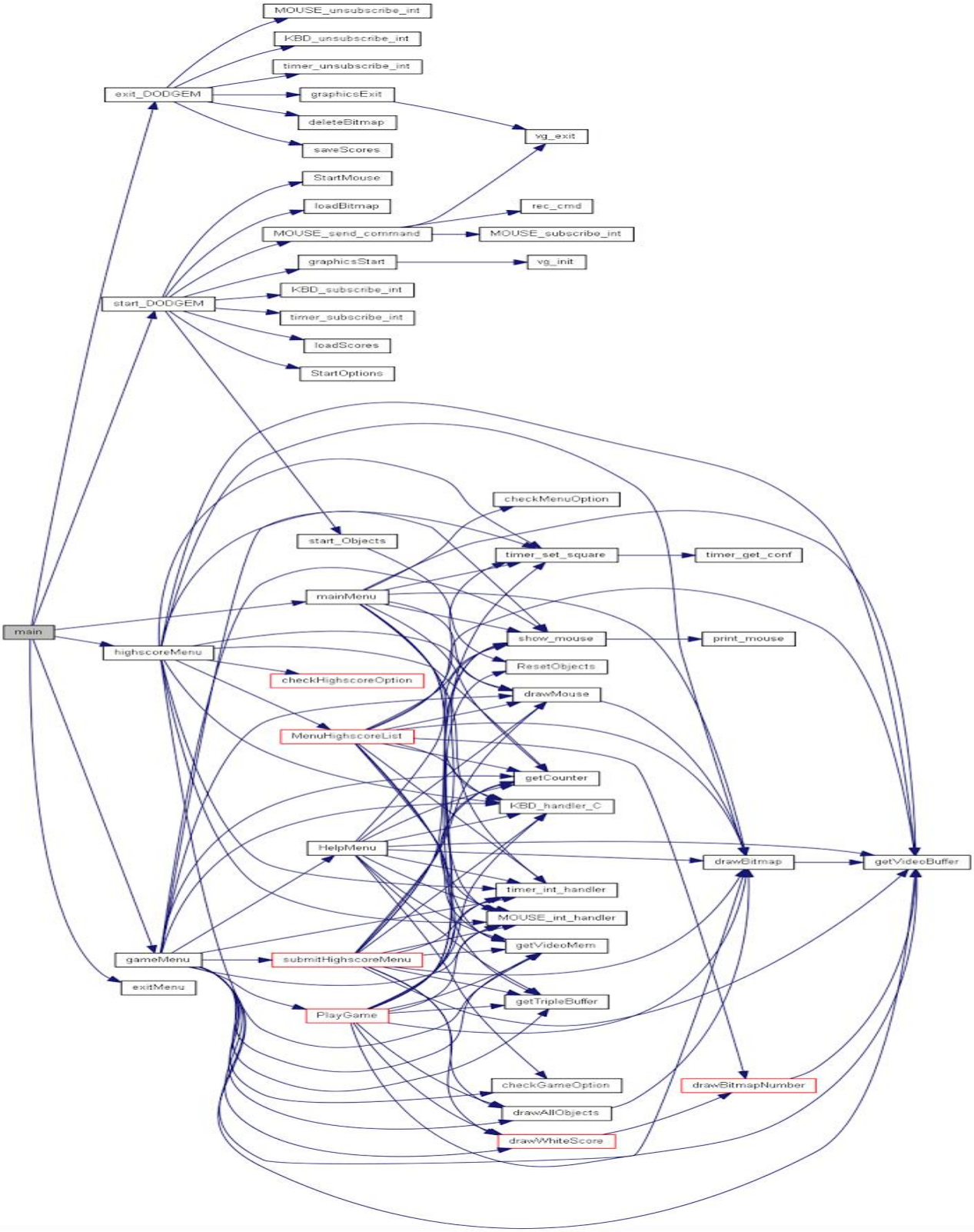


Ilustração 9: Menu de Ajuda

Call Graph:


MOUSE_unsubscribe_int
KBD_unsubscribe_int
timer_unsubscribe_int
exit_DODGEM
graphicsExit
deleteBitmap
saveScores
vg_exit
StartMouse
loadBitmap
rec_cmd
MOUSE_send_command
MOUSE_subscribe_int
graphicsStart
vg_init
KBD_subscribe_int
start_DODGEM
timer_subscribe_int
loadScores
StartOptions
checkMenuOption
start_Objects
timer_set_square
timer_get_conf
mainMenu
main
show_mouse
print_mouse
highscoreMenu
ResetObjects
checkHighscoreOption
drawMouse
MenuHighscoreList
getCounter
KBD_handler_C
HelpMenu
drawBitmap
getVideoBuffer
timer_int_handler
MOUSE_int_handler
getVideoMem
gameMenu
submitHighscoreMenu
exitMenu
getTripleBuffer
PlayGame
checkGameOption
drawBitmapNumber
drawAllObjects
drawWhiteScore

Execução do projeto e outros aspectos relevantes:

Para a execução do projeto usamos o script disponibilizado pelo estudante Henrique Ferrolho. Para usar o script é necessário estar na pasta do projeto e executar o comando sh install.sh, para copiar o serviço para a pasta /etc/system.conf.d seguido de sh compile.sh para compilar e por fim sh run.sh para executar.

É necessário também copiar a biblioteca liblm.a dos ficheiros do redmine para a pasta code e alterar as macros de "utilities.h" para o caminho dos ficheiros repetivos caso necessário.

Outros aspetos:

- O código por nós utilizado para desenhar no buffer é baseado no código disponibilizado pelo estudante Henrique Ferrolho, no entanto primeiro percebemos o código disponibilizado e depois adaptamos à nossa maneira.
- Durante a execução do programa reparamos que em certas alturas o jogo deixa de responder e o processo termina. Tentamos perceber porquê, mas sem sucesso, no entanto é algo que achamos não ter grande importância, uma vez que isto só acontece muito raramente e o programa, de resto, funciona perfeitamente.
- Os caminhos das imagens e do ficheiro de scores encontram-se definidos no ficheiro "utilities.h".

Avaliação:

Auto-avaliação:

Participantes	**Daniel Reis**	**João Baião**
Participação	0.5	0.5
Contribuição	0.5	0.5

O trabalho foi distribuido de forma igual pelos dois, sendo que ambos temos conhecimento de tudo o que está presente no código do programa, da maneira e dos porquês de estar como está.

Avaliação da disciplina:

Os maus:

- O material de estudo disponibilizado é muito confuso e pouco explicativo.
- A maneira como as teóricas são dadas não cativam os estudantes como se calhar deviam.
- O pouco tempo e a carga de trabalho que a cadeira requer são um pouco exagerados.
- A diferença na data de entregas dos labs acaba por prejudicar algumas turmas, como foi o caso da nossa que tinha aulas à segunda feira e ainda por cima as teóricas eram às terças.

Os bons:

- É uma cadeira cativante pelo facto de possibilitar aos estudantes trabalharem em algo que dá gosto fazer, que é programar em alto nível e em alguns casos fazer jogos, que é algo divertido até de se fazer.
- O projeto.

Conclusão

Com a elaboração deste trabalho, podemos aprofundar os nossos conhecimentos de programação em C e também compreender como ocorrem interrupções de periféricos.

O desenvolvimento deste trabalho não apresentou muitas dificuldades e no geral, estamos bastantes satisfeitos com o resultado final pois conseguimos implementar tudo aquilo que inicialmente tínhamos planeado.

No final, ficamos com um jogo bastante fluido e com uma implementação bastante eficaz.